REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE MAY 1995	3. REPORT TYPE AND DATES COVERED
4. TITLE AND SUBTITLE An ABSTRACT Representation For Model-Based Computer Vision		5. FUNDING NUMBERS
6. AUTHOR(S) Sheila Benfield Banks		
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFIT Students Attending: Clemson University		8. PERFORMING ORGANIZATION REPORT NUMBER AFIT/CI/CIA 95-008 D
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DEPARTNEMT OF THE AIR FORCE AFIT/CI 2950 P STREET, BDLG 125 WRIGHT-PATTERSON AFB OH 45433-7765		10. SPONSORING/MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES		
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for Public Release IAW AFR 190-1 Distribution Unlimited BRIAN D. GAUTHIER, MSgt, USAF Chief Administration		12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

14. SUBJECT TERMS			15. NUMBER OF PAGES 118
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT

ABSTRACT

The current work presented is research into a general and flexible representation technique for model-based computer vision. This abstract representation integrates various sources of knowledge within model-based vision: functional, geometric, and relational; and provides a representation to express image data, image extracted parameters, and model parameters. The abstract representation is based upon the notion of feature structures as derived from linguistic applications of unification grammars and the manipulation technique of unification. The representation of feature structures and the manipulation technique of unification are combined into a lattice structure that is partially ordered by the subsumption relationship. These aspects are explored as the backbone of the abstract representation of the unification grammar application to model-based computer vision. Promising aspects of accomplishing parallel unification and unifying solution search lattices in parallel are discussed. An evaluation of the major results of this work and a discussion on areas for future research conclude this dissertation.

AN ABSTRACT REPRESENTATION FOR MODEL-BASED COMPUTER VISION

A Dissertation

Presented to

the Graduate School of

Clemson University

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

Computer Engineering

by

Sheila Benfield Banks

May 1995

DEDICATION

Dedicated To

My Family

ACKNOWLEDGEMENTS

I would like to express my appreciation to my advisor, Dr. Robert J. Schalkoff, for his creative guidance and patience during my studies. I would also like to thank my committee members Dr. James F. Leathrum, Dr. A. Jefferson Offutt, and Dr. Walter Ligon for their advice.

I would like to give special thanks to the United States Air Force who provided the opportunity and funding for my studies at Clemson University. Also, thanks to Dr. Kathleen A. Liburdy for her insights into the original application of unification grammars to object classification.

Finally, I would like to express my gratitude to my husband for his unending support of my efforts and to my parents for instilling in me the love and value of learning.

TABLE OF CONTENTS

Table of Contents (Continued)

LIST OF FIGURES

List of Figures (Continued)

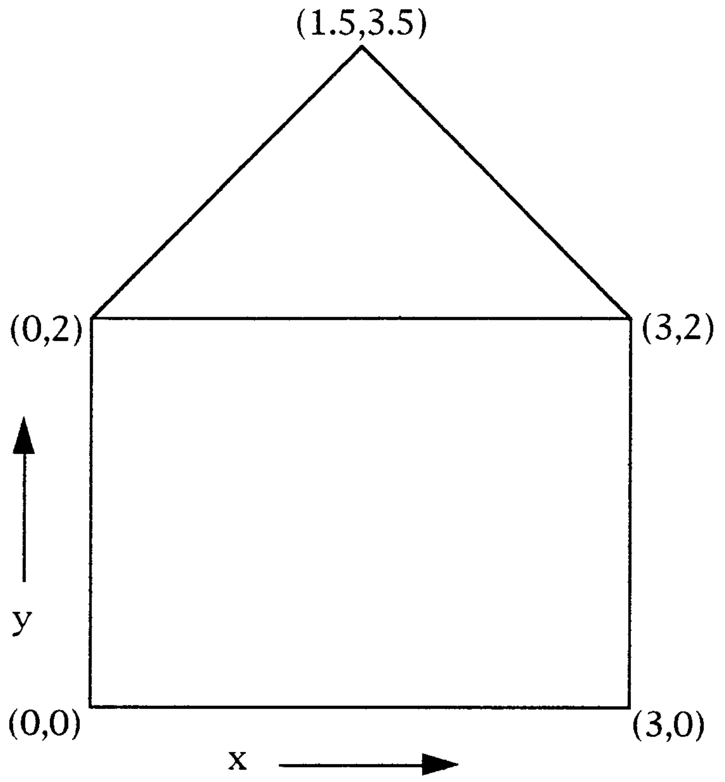
CHAPTER 1
INTRODUCTION

In general, artificial intelligence (AI) is a field of study that seeks to explain and emulate intelligent behavior in terms of computational processes. Any computation requires a representation of some entity as well as a procedure for manipulation. Representation and manipulation are key elements of AI and knowledge cannot be manipulated unless it is adequately represented. Therefore, on an engineering level, AI is centered around generating representations and procedures that automatically or autonomously solve problems that were previously only solvable by humans.

Computer vision, an area of AI research, is concerned with developing systems that, through algorithms and computation, produce the same answer as a human when asked "What do you see?" The first concern of a computer vision system, as an AI system, must be in developing an adequate representation of knowledge required for computer vision and the manipulation techniques to allow use and interpretation of the available knowledge. Model-based computer vision takes an approach that describes all the available information about an entity in terms of a world model and tries to form a match between the represented model and information extractable from an image that is represented to the computer. From these matches, objects in the image may be identified or the entire image scene may be interpreted. Scene interpretation, which corresponds to a human viewing a cake, candles, and balloons and determining the image is of a birthday party, is one goal of a computer vision system. The problems in model-based computer vision center around the appropriate representation of the world models and image data and the corresponding manipulation technique to match extracted image data and the represented world model. The computer vision system then uses these matches to draw conclusions about the content of the input image.

A robust computer vision system will require a representation that possesses the ability to include all types of information about a modelled entity. The information contained in the model might be geometrical, functional, relational, or any quantifiable or qualifiable aspect about the entity to be included in the vision system. The vision system might be required to find an instance of a specific entity in an image, find all instances of a generic entity in the image, or correlate the labelled entities into a coherent scene. These tasks cannot be performed without many different types of information available to a computational mechanism.

Most current computer vision systems concentrated solely on a single type of model within the system, for example, geometric models. Using a single model restricts the type of information available in the model and also the type of information that can be extracted, matched, and derived from the image. This single model restriction often leads to a brittle and over-specialized vision system. The problems in developing an abstract representation and the corresponding manipulation technique are derived from the lack of ability to span the representation from the inherently numerical extracted image parameters to the level of abstract semantic information required to perform scene interpretation. Figure 1 illustrates this representation gap between the basically numerical image data and different world model representations of the same image data. The goal of this work was to investigate an abstract representation within model-based computer vision that will span the representational gaps of different types of



(a) Image Data of the Boundary of a House

object: house
top_component: triangle
bottom_component: box
relation: on_top_of

line segments:

line1: end1: (0,0)
end2: (3,0)

line2: end1: (0,0)
end2: (0,2)

line3: end1: (3,0)
end2: (3,2)

line4: end1: (0,2)
end2: (1.5,3.5)

line5: end1: (1.5,3.5)
end2: (3,2)

(b) Two Possible World Models of the House

Figure 1. Representational Gap of World Models and Image Data

world models and the image data and to demonstrate the possibility of a robust and intelligent computer vision system resulting from the use of the abstract representation and its associated manipulation technique.

1.1 Problem Specification

The major problem in current model-based computer vision systems is the lack of generality and scalability. This problem is a direct result of the use of a single modeling technique; whether the modeling technique be geometrical, functional, or relational. Systems that employ only one of the previous modeling techniques present both strengths and weaknesses. However, the weaknesses of a particular modeling technique usually lead to brittle system behavior and not the robust system behavior desired of a general computer vision system.

Utilizing a single model also raises additional issues for a model-based computer vision system. One of these issues is how to choose a single model that is the best for the developing vision system. Should a geometric model be used and should this model be in 2-D, 21/2-D, or 3-D? Maybe the model should be functional a take advantage of the ability to group classes of objects without individual geometric determination. Or maybe the model should take advantage of relational aspects such as adjacency and relative positions. Another issue with a single modeling technique arises once the modeling technique is chosen. The issue then becomes how to determine whether the extracted image data fits the chosen model and may, therefore, be utilized in the current vision system. Once the extracted image data can be used to fit against the chosen modeling technique, the issue is at what level of resolution should the model be to provide the best use of the image data for matching. Therefore, in current vision systems, one modeling technique is typically chosen with the goal to maximize benefits of the model and minimize the shortcomings of its use.

The general problem to be addressed is the following: How can a general modeling technique, or an abstract model representation, fuse various modeling techniques and also relate the image-derived features with the model features. The solution to this problem lies in a flexible abstract representation and the ability to manipulate the representation for use in a computer vision system. The central questions to be answered in each of these areas are addressed next.

1.1.1 Abstract Representation

An abstract representation is a scheme used to capture the essential elements, which express the relevant aspects of the class of objects for which a model (and consequently a representation) is desired, within a problem domain. The ability to use a proposed abstract representation lies in its representational flexibility. The abstract representation must be able to accommodate the different types of required information: geometric (2-D and 3-D), functional, relational, uncertainty, multiple resolutions, and multiple representational levels. The abstract representation must be able to meld the various types of information into a coherent general model for model-based reasoning within the overall vision system. The abstract representation must also be able to represent the image-derived data, including image features and image-based relational information. In this manner, the extracted image features may be directly comparable with the features represented in the general model.

1.1.2 Manipulation Technique

The abstract representation and the manipulation technique are not separate issues, but closely interconnected. The choice of a representation strongly effects the appropriate manipulation techniques of the representation. An appropriate manipulation technique must provide the basis for computer manipulation of the abstract representation and also the additional methods required for a robust computer vision system. These methods include the ability to handle incompletely observed information and default values; the ability to incorporate different control strategies and search techniques; the ability to promote hypothesis generation; and the ability to control computational complexity.

1.2 Organization of Remaining Discussions

The abstract representation of the current work and the appropriate manipulation technique for the representation are discussed in Chapters 3 and 4 respectively. Prior to this discussion, Chapter 2 provides the background material necessary to relate the current work to other areas in model-based computer vision and also background material closely related to this research from other related fields. Experiments to illustrate the operation of the abstract representation in a representative setting are presented in Chapter 5. Chapter 6 focuses on possible parallel representations of the chosen manipulation technique and, finally, Chapter 7 concludes the discussion of the current work and also presents areas for future expansion of the current research.

CHAPTER 2
BACKGROUND

This section discusses the major areas of research that directly influence the current work. There have been years of research into model-based computer vision using models based on geometric, relational, or functional models. However, there have been few, if any, research attempts into model representations that integrate the benefits of both high-level functional models with geometric and relational aspects. The current representation technique also relies heavily on the use of lattices for visualization and manipulation. Therefore, a basic understanding of lattice theory was necessary in the current research. Finally, the research into methods for manipulation of the models in a vision system also influences the current work. Model unification is a manipulation concept that overrides much of model-based vision research and is the focus of the investigation directly related to this dissertation.

2.1 Model-Based Representations

Almost every model-based computer vision or object classification system has its own unique method of representing the model in use. By far the most prevalent in the literature is the representation of the geometrical properties of a system model. Description of the functional properties for a model is a much newer modeling concept and is becoming more accepted as a modeling technique.

2.1.1 Geometrical Modeling

Models based on geometric properties of an object's visible surfaces or silhouette are commonly used because they describe objects in terms of their constituent shape features. Alternative methods for representing object models using 2-D, 21/2-D, and 3-D shape features are all present in the literature. Excellent survey articles aimed at general vision systems and their components and requirements are presented by Binford [1], Chin and Dyer [2], and Wallace [3]. These articles compare many current vision systems and vision approaches and note the strengths and deficiencies of each approach. Of particular interest is the ACRONYM system by Brooks [4, 1]. This system has its basis in geometric modeling of objects and also uses extensive shape information for additional model information. ACRONYM's interpretation is generic with respect to observation, that is, the interpretation is insensitive to viewpoint and flexible with varied sensor inputs. The objects that the system can recognize are stored in terms of primitive component parts called generalized cylinders that are assembled together according to geometric rules. ACRONYM uses geometric reasoning techniques to predict object features (subparts) that can be seen or not seen from a particular perspective. The system generates hypotheses about the observability of object features and is able to prove or disprove them based on the actual features extracted from the image. ACRONYM continues to generate hypotheses and gain information about the detected features in a backward chaining fashion until it is able to recognize the objects of which it has knowledge.

Many interesting modeling techniques abound in the model-based vision arena. A relatively new geometric modeling technique is termed "CAD-based vision". This modeling representation uses the exact geometric descriptions that are available from a CAD system [5]. With a CAD-based vision system, a unique 3-D model is stored for each object that the system is able to recognize. Another geometric modeling technique is the aspect representation [6]. This representation is a continuous, viewer-centered representation for 3-D objects. The representation is continuous in the sense that the appearance of objects can be represented for all viewpoints rather than a discrete set of viewpoints and is viewer-centered as opposed to object-centered in the sense that the appearance of an object is represented as a function of viewpoint. The aspect representation makes explicit visual events that occur from various viewpoints in a scene. By making such events explicit, the representation is useful in dealing with problems that depend upon knowing or predicting the appearance of an object from many viewpoints, determining a viewpoint at which a specific visual event occurs, and finding ranges of viewpoints with particular visual characteristics.

Motivated by understanding of the human visual system, the method of qualitative structural models of geometry is receiving increased attention [7]. Rather than representing the geometry of the object with analytic models, abstract qualities are used to model important aspects of the object. Biederman's theory of image understanding is referred to as "recognition-by-components". The primitive components of recognition can be represented with a qualitative description scheme indicating the types of regularities displayed. Colinearity, symmetry, and parallelism are typical qualities used to model the object in this theory. A vision system developed by Topa [8] employs this type of qualitative representation for the purpose of recognizing objects.

2.1.2 Functional Modeling

Instances of reasoning using only geometric models often lead to an oversimplified vision system that cannot be used for a general vision system on groups of objects in the same class, but with varying geometric properties. The purpose of functional modeling is to define an object category in terms of the functional properties shared by all objects in the category. This may be done without explicit geometric or structural modeling and the use of a purely function-based definition of an object category model [9, 10, 11, 12]. An aspect of functional modeling is to reason about the relation between shape and function and use the functional concepts that are then defined in terms of the shape representation. All of these works point to the argument that a geometric model is typically restricted to a small range of instances of a much larger class of objects and that a geometric model is not at the proper level of abstraction to support the desired level of reasoning in a general computer vision system. These works do not state, however, that a system employing geometric models cannot accomplish high-level computer vision tasks.

Work in semantic modeling [10], a type of functional modeling, is the basis for the current research. The semantic model is described as the ability to model the essence of an object. For example, a semantic model of an object such as a table might include 'has supporting base', while a geometric model would actually describe the physical properties of a particular base (e.g., four cylindrical legs). The single semantic description may correspond to many geometric descriptions. The main issues of Liburdy's research centered on bridging the large gap between the semantics of this type of model and currently derivable image features (primarily geometric). Specific goals included the development and implementation of a generic object modelling paradigm that incorporated semantics, and a methodology for deriving an image description suitable for interaction with high level semantic models. This work developed a system of object classification using unification-based grammars to relate image data and object models. Unification grammars encode knowledge as a declarative specification of constraints rather that a procedural definition of derivations. Within the unification-based grammar framework, a complex data structure referred to as a feature structure [13, 14], or functional description [15], encodes information from various sources as feature/value pairs. Feature structures resemble first-order logic terms but have several restrictions lifted [13]. A feature structure may be viewed as a partial function from features to their values [14]. For instance, a function mapping the feature object onto the value house and mapping viewpoint onto side would be represented in a feature structure notation as in Figure 2. Feature structures may be atomic (Figure 2) or complex (Figure 3). Complex feature structures contain partial functions from features to values that are themselves feature structures.

Feature structures resemble first-order logic terms but have several important distinctions [13]. The first difference is that feature substructures are labeled symbolically, not inferred by argument position. First-order terms require strict ordering of information; feature structures lift this restriction by labeling the information and, therefore, the information in the feature structures must be considered in an order independent fashion. The second distinction is that fixed arity is not required. Feature structures may represent partial information. Feature structures containing partial information may be combined into larger structures, assuming no conflict of information occurs. The next important difference is that the distinction between function and argument is removed. This aspect of feature structures is termed the demoted functor. In first-order terms, the functor (function

$$\begin{bmatrix} \text{object: house} \\ \text{viewpoint: side} \end{bmatrix}$$

Figure 2. Sample Feature Structure

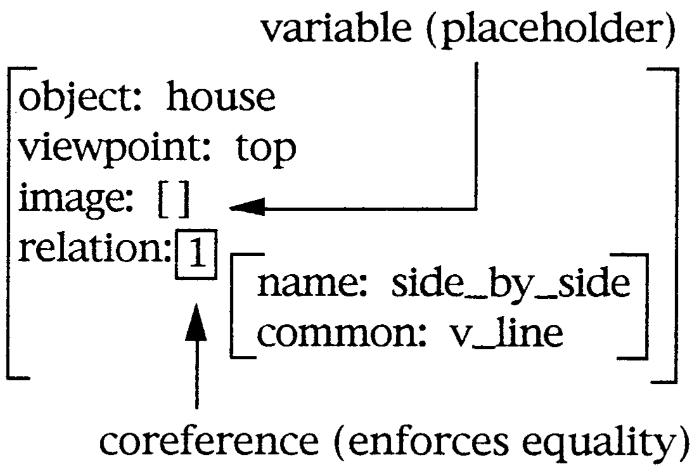


Figure 3. Complex Feature Structure - Variables and Coreference

symbol that begins the term) has a special significance. In feature structures, all information has equal status. The last distinction is that variables and coreference are treated separately (Figure 3). Variables in first-order terms serve two purposes: place holders for future instantiations and enforcement of equality constraints among different parts of the term. In feature structures, these two purposes are served by different means. Coreference, or marking attributes with the same value label, forces the attributes to share the same value. The feature structure variable [ ] does not enforce equality constraints, but simply serves as a placeholder that may unify with any other feature structure. The feature structure is the basis for the current abstract representation and its application to the computer vision representation problem will be discussed further in Section 3.1.

2.2 Lattices

The current representation technique also relies heavily on the use of lattices for visualization and manipulation. A basic understanding of lattice theory was necessary and an introduction to lattices is presented here.

2.2.1 Lattice Definition

The definition of a *partial order* is necessary before a lattice can be defined. A relation R on the set L is a *partial order* if R is reflexive, antisymmetric, and transitive [16]. The symbol $\leq$, rather the R, is often used to denote a partial order. Therefore, a *lattice* is a *partially ordered set,* or *poset,* $(L,\leq)$ in which every subset $\{a,b\}$ consisting of two elements has a *least upper bound* (LUB) and a *greatest lower bound* (GLB). The LUB of two elements a and b is called the *join* of a and b, denoted by JOIN($\{a,b\}$). Likewise, the GLB of two elements a and b is called the *meet* of a and b, denoted by MEET($\{a,b\}$).

A nonempty subset S of a lattice L is called a *sublattice* of L if both the JOIN($\{a,b\}$) and MEET($\{a,b\}$) are contained within the sublattice S whenever a and b are elements of S. A lattice L is a special type of lattice, a bounded lattice, if it has a greatest element I and a least element O. Figure 4 shows a simplified bounded lattice with a greatest element, least element, join, meet, and sublattice identified.

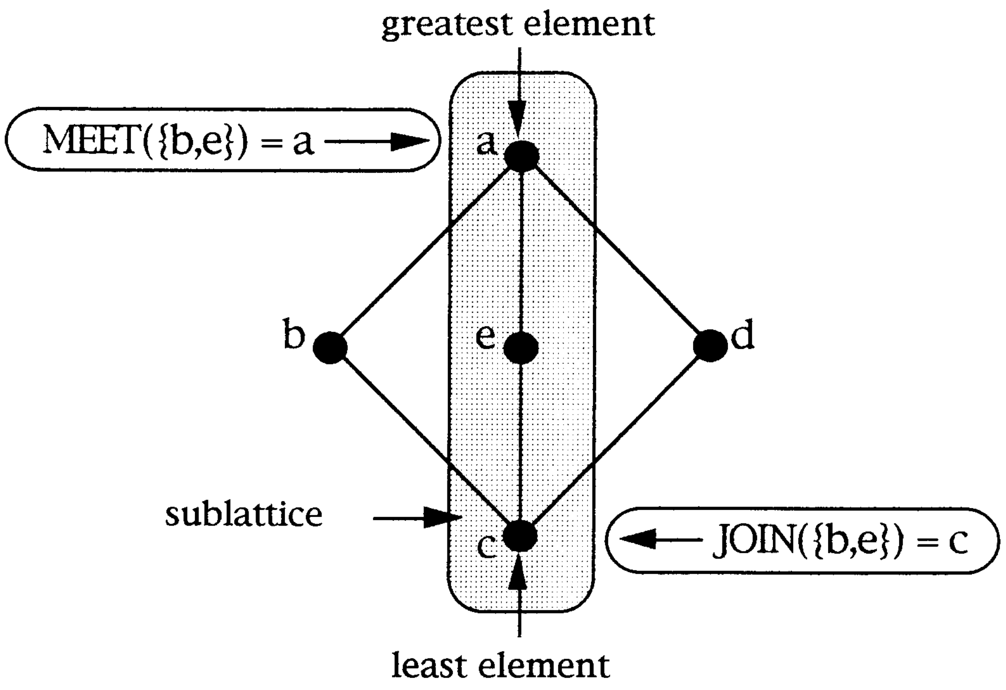


Figure 4. Example Lattice

2.2.2 Properties of Lattices

These three theorems identify the basic properties of lattices that are useful in the current work.

Theorem 1 Let L be a lattice. Then for every a and b in L:

(a) JOIN($\{a,b\}$) = b if and only if $a \leq b$.

(b) MEET($\{a,b\}$) = a if and only if $a \leq b$.

(c) MEET($\{a,b\}$) = a if and only if JOIN($\{a,b\}$) = b.

Theorem 2 Let L be a lattice. Then

1. Idempotent Property
   - (a) JOIN($\{a,a\}$) = a
   - (b) MEET($\{a,a\}$) = a
2. Commutative Property
   - (a) JOIN($\{a,b\}$) = JOIN($\{b,a\}$)
   - (b) MEET($\{a,b\}$) = MEET($\{b,a\}$)
3. Associative Property
   - (a) JOIN($\{a$,JOIN($\{b,c\}$)$\}$) = JOIN($\{$JOIN($\{a,b\}$),$c\}$)
   - (b) MEET($\{a$,MEET($\{b,c\}$)$\}$) = MEET($\{$MEET($\{a,b\}$),$c\}$)
4. Absorption Property
   - (a) JOIN($\{a$,JOIN($\{a,b$)$\}$) = a
   - (b) MEET($\{a$,MEET($\{a,b$)$\}$) = a

Theorem 3 Let L be a lattice. Then for every a, b, and c in L:

1. If $a \leq b$, then
   - (a) JOIN($\{a,c\}$) $\leq$ JOIN($\{b,c\}$)
   - (b) MEET($\{a,c\}$) $\leq$ MEET($\{b,c\}$)
2. $a \leq c$ and $b \leq c$ if and only if JOIN($\{a,b\}$) $\leq c$.
3. $c \leq a$ and $c \leq b$ if and only if $c \leq$ MEET($\{a,b\}$).
4. If $a \leq b$ and $c \leq d$ then
   - (a) JOIN($\{a,c\}$) $\leq$ JOIN($\{b,d\}$)
   - (b) MEET($\{a,c\}$) $\leq$ MEET($\{b,d\}$)

2.3 Manipulation Process

Given a set of models that describes all aspects of all parts to be recognized, the process of model-based recognition consists of matching features, which include geometrical, syntactical and relational information, extracted from a given input image with those of the models. The general problem of matching may be regarded as finding a set of features in the given image that approximately matches one model's features. The choice of a matching process is highly dependent on the type of model used for object representation. Unification, a common matching method in model-based computer vision [17], is the single operation used to manipulate the feature structures [14] and is the basis for manipulation in the current research.

2.3.1 Unification

Unification has been studied in a number of computer-related fields such as natural language processing, logic programming, theorem proving, computational complexity, and computability theory [13]. Abstractly stated, the unification problem is the following: Given two descriptions x and y, can we find an object z the fits both descriptions? Knight states the unification problem is most often given in the following context: Given two terms of logic built up from function symbols, variables, and constants, is there a substitution of terms for variables that will make the two terms identical? The nature of this unifying substitution, as well as the means of computing the substitution, makes up the study of unification. Unification of first-order terms is the basis for the Prolog unification mechanism.

2.3.2 Unification and Feature Structures

Liburdy [10], Knight [13], Shieber [14], and Kay [15] all discuss unification as the sole-information combining operation of feature structures. Unification of feature structures results in a new structure that contains all of the information from the two structures to be unified, assuming no attribute/value pairs contain conflicting information (Figure 5). Unification of feature structures can be defined with the introduction of the partial ordering relation *subsumption.* Intuitively, a feature structure *D* *subsumes* a feature structure *D'* (denoted $D \sqsubseteq D'$) if *D* contains a subset of the information in *D'*. More precisely, a complex feature structure *D* subsumes a complex feature structure *D'* if and only if $D(l) \sqsubseteq D'(l)$ for all *l* contained in the *dom(D) and D'(p) = D'(q) for all paths p and q such the D(p) = D(q)* [14]. The notation *D(l)* denotes the value associated with the feature *l* in the feature structure *D* and the *domain* of a feature structure *D*, *dom(D)*, consists of the features in the feature structure *D*. For example, in Figure 2 *D(object) = house* and *Dom(D) = {object, viewpoint}*. A feature structure with an empty domain is often called an empty feature structure or a variable. A path in a feature structure is a sequence of features that can be used to pick out a particular subpart of a feature structure by repeated application and is denoted *D(p)*. In Figure 3, an example path (sequence of features) is *D(<relation common>) = v_line.* In addition, an atomic feature structure neither subsumes nor is subsumed by a different atomic feature structure. Variables subsume all other feature structures, atomic or complex, because, as the trivial case, they contain no information at all. As a partial ordering relation, subsumption is reflexive, antisymmetric, and transitive. The ordering imposed by subsumption provides a framework or lattice for characterizing unification [13]. Figure 6 shows a portion of a lattice containing simple feature structures augmented with two special terms call top ($\top$) and bottom ($\perp$); making the lattice a bounded lattice. Unification corresponds to finding the greatest lower bound (or meet) of two feature structures in the lattice. Figure 7 shows the unification of feature structures within a partial lattice of feature structures (arrows point to the unifying feature structure of two feature structures). The top of the lattice ($\top$) is also called the *universal variable* because it is the trivial case of containing no information and subsumes all other feature structures in the lattice. The bottom of the lattice ($\perp$), to which all pairs of terms can unify, represents inconsistency. If the greatest lower bound of two terms is the bottom ($\perp$), then they are not unifiable.

$$\begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix} \sqcup \begin{bmatrix}\text{box: [ ]} & \\ \text{triangle: [ ]} & \\ \text{relation:} & \begin{bmatrix}\text{name: on\_top\_of}\\ \text{top\_component: triangle}\\ \text{bottom\_component: box}\end{bmatrix}\end{bmatrix}$$

$$\longrightarrow \begin{bmatrix}\text{object: house} & \\ \text{viewpoint: front} & \\ \text{box: [ ]} & \\ \text{triangle: [ ]} & \\ \text{relation:} & \begin{bmatrix}\text{name: on\_top\_of}\\ \text{top\_component: triangle}\\ \text{bottom\_component: box}\end{bmatrix}\end{bmatrix}$$

Figure 5. Unification of Two Feature Structures

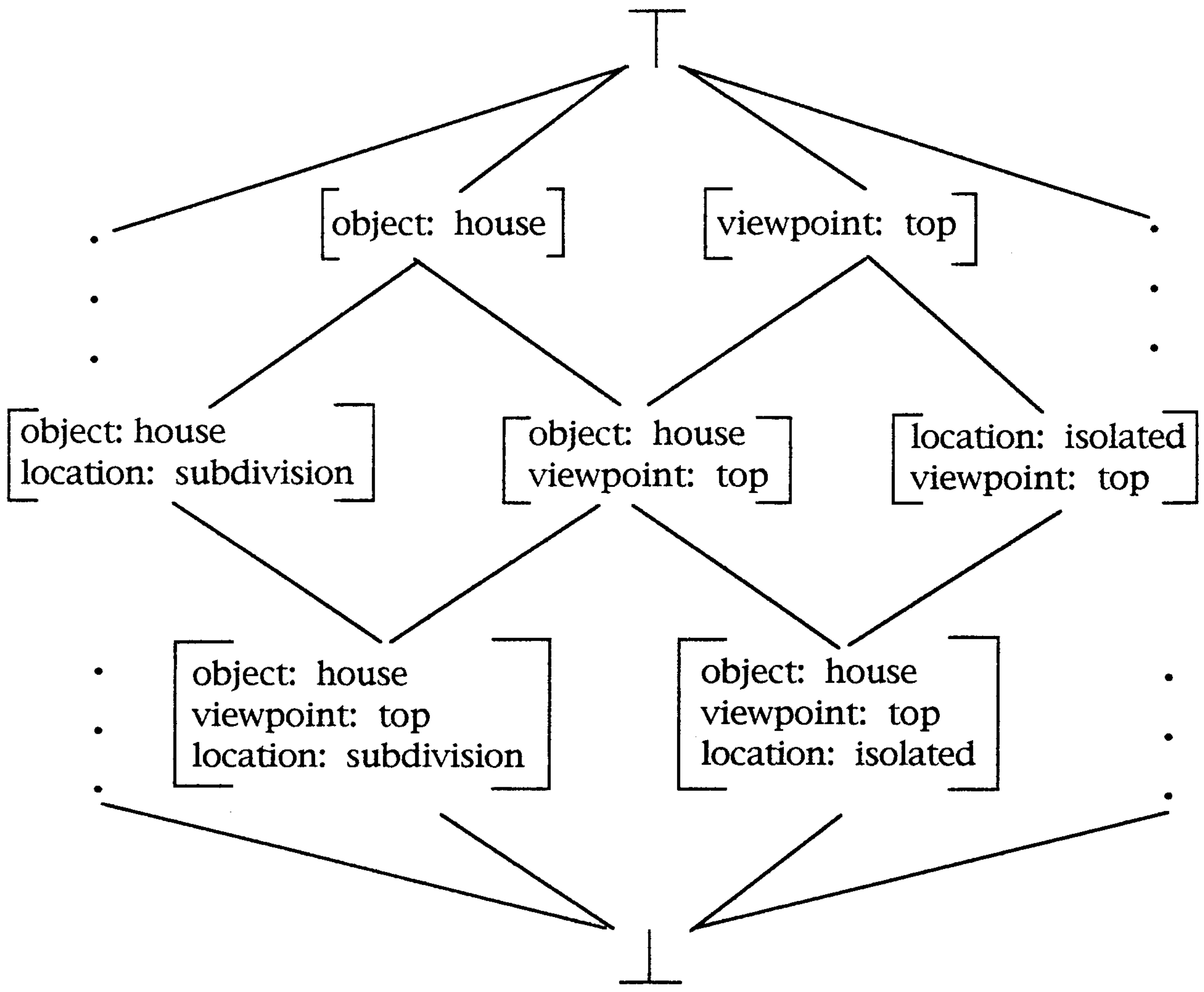

⊤
object: house
viewpoint: top
object: house location: subdivision
object: house viewpoint: top
location: isolated viewpoint: top
object: house viewpoint: top location: subdivision
object: house viewpoint: top location: isolated
⊥

Figure 6. A Portion of a Lattice of Feature Structures

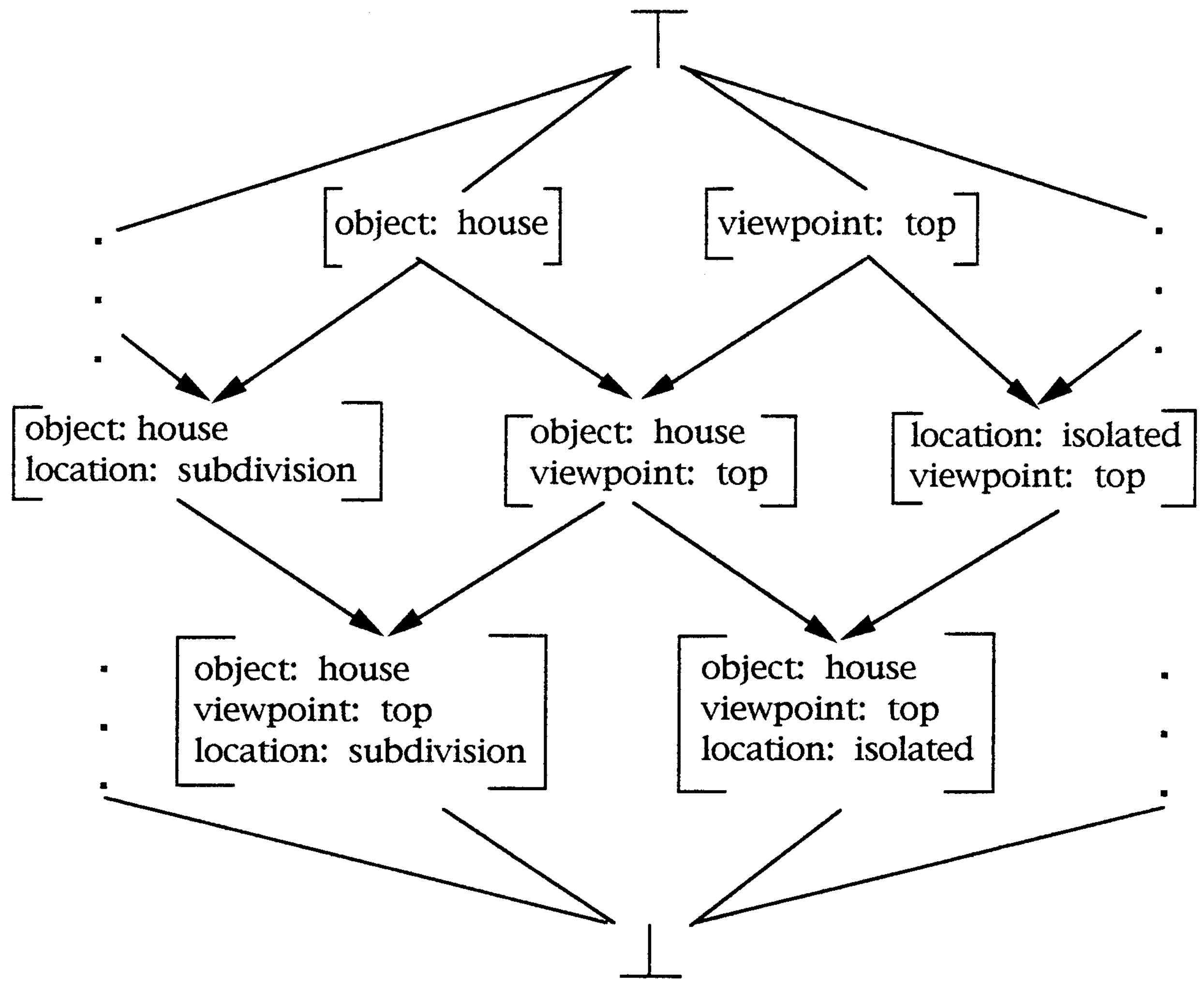

⊤
[object: house]
[viewpoint: top]
[object: house location: subdivision]
[object: house viewpoint: top]
[location: isolated viewpoint: top]
[object: house viewpoint: top location: subdivision]
[object: house viewpoint: top location: isolated]
⊥

Figure 7. Unification in a Lattice of Feature Structures

CHAPTER 3
ABSTRACT DATA REPRESENTATION

The representation of feature structures as derived from linguistic applications of unification grammars and an introduction to basic lattices was discussed previously. These two essential components form the research foundation for the abstract data representation presented here.

3.1 The Feature Structure Representation

The abstract representation is based upon the feature structure as discussed in Liburdy [10], Knight [13], and Shieber [14]. Within the current work there are additions to the basic feature structure. These additions include disjunctive features, range-valued and multi-valued features, and functional processing features. A discussion of why the feature structure representation is important in the computer vision problem is also addressed at this point.

3.1.1 Disjunctive Features

Disjunctive features are characterized by a feature that may be specified by a set of values, at least one of which must be its true value. Disjunctive features are often seen as a logical construct that is useful in describing feature generalizations within the feature structure representation. There are many useful classes of feature structures the cannot by modeled by using only a single partial feature structure [18]. Disjunctive features are a way to model these classes of feature structures and still maintain the simplicity of the representation. Disjunctive features are of great utility in many linguistic areas [13, 19]. The most prominent use of disjunction is in describing linguistic generalizations, some of which cannot be described by using only a single conjunctive feature structure [18]. In order to characterize these linguistic generalizations in terms of conjunctive feature structures, a set of structures needs to be given. This set of conjunctive feature structures corresponds to the disjuncts in what is called the *disjunctive normal form* (DNF) of the description. The DNF is simply a set of conjunctive feature structures that can be interpreted with the same meaning as the single disjunctive feature structure. However, it is well known that the DNF expansion of a description may be exponentially larger than the original disjunctive description, so it is possible to characterize such disjunctive features more succinctly at the level of disjunctive descriptions than at the level of conjunctive feature structures [18].

The use of disjunctive features to represent image information is a new application of this concept. A sample of an computer vision based disjunctive feature is shown in Figure 8. The curly brackets ('{ }') indicate disjunction. The disjunctive feature's value may be any one of the structures inside the brackets. All feature value possibilities are maintained when performing unification. The unification process of disjunctive features will be discussed further in Section 4.1.2. Disjunctive features are useful in the abstract data representation for the implementation of many of the numerical features within the system. The image data, image data transforms, boundary data, and line data are types of feature information represented using disjunctive features.

$$\begin{bmatrix} \text{box:} \left\{ \begin{matrix} \begin{bmatrix} \text{joinpoints:} & \begin{bmatrix} \text{x1: } 2 \\ \text{y1: } 0 \\ \text{x2: } 2 \\ \text{y2: } 4 \\ \text{x3: } 4 \\ \text{y3: } 0 \\ \text{x4: } 4 \\ \text{y4: } 4 \end{bmatrix} \end{bmatrix} \\ \begin{bmatrix} \text{joinpoints:} & \begin{bmatrix} \text{x1: } 6 \\ \text{y1: } 0 \\ \text{x2: } 6 \\ \text{y2: } 4 \\ \text{x3: } 8 \\ \text{y3: } 0 \\ \text{x4: } 8 \\ \text{y4: } 4 \end{bmatrix} \end{bmatrix} \end{matrix} \right\} \end{bmatrix}$$

Figure 8. Disjunctive Feature Structure

3.1.2 Range-Valued and Multi-Valued Features

Another extension to the basic feature structure used in the current work was range-valued and multi-valued features. The features defined in terms of range-valued and multi-valued may take on more than one value. An example of a range-valued feature is shown in Figure 9. To distinguish range-valued from multi-valued, the range-valued feature is defined by a range of values, usually numerical in nature. A multi-valued feature may be defined by any set of values, not necessarily related as in range-valued features. The unification process is the same for both range-valued and multi-valued features and will be discussed in Section 4.1.3. Within the current work range-valued features were primarily used to allow for noisy image data, uncertainty in the features derived from the image, and in feature structure rules as a certain number for things that may be present in an image.

$$\begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } (5, \ldots, 20) \\ \text{viewpoint: top} \end{bmatrix} \\ \text{image\_content: neighborhood} & \end{bmatrix}$$

Figure 9. Range-Valued Feature

3.1.3 Functional Processing Features

The last major extension to the basic feature structure representation is the addition of functional processing as the value of a feature. This type of feature value will be computed as the system is running instead of being a static value from the image data or derived by rule application during unification. Most information represented in this manner is information to be calculated from the initial image data. Since the image data is contained in the computer vision system in the current feature structure, the image data may be accessed at any time throughout system operation. This is unlike many high-level computer vision systems who processes the numerical image data into the desired format for the modelling technique used by the system and then discard the original image data. Because of this, the image data is not available for further image parameter extraction later in the image determination cycle or to recalculate the processed image data using a different image processing algorithm. In addition, most data processing algorithms that are applied to the image data itself are numerical in nature and, for the most part, have been used for many years in many types of image processing packages. To use unification to calculate this type of information from the initial image data was a drain on the solution time. Therefore, the feature structure was allowed to have a functional processing value that, as a side-effect of unification, calculates a value for the present feature using the indicated function. Figure 10 shows a feature structure containing a functional processing value for the image_boundary feature. When the feature structure in Figure 10 is unified with any other feature structure in the computer vision system, the value of the image_boundary feature is calculated at that time using the specified function boundary_function. The value returned by the function then replaces the functional processing value in the current feature structure if unification is successful. In this manner, many different processing functions may be used as needed in the system. The need to pre-compute all the types of information from the initial image data that may be required as the system progresses is obviated and only the information that is actually required by the current system run to determine the content of the current image is calculated. Unlike many current computer vision systems, the original image data is also maintained in the feature structure and may be used at any time by any processing function to obtain a needed numerical quantity to aid in the image determination.

$$\left[\begin{array}{ll} \text{has\_image:} & \left[\begin{array}{l} \text{value: } [\,] \\ \text{input: yes} \\ \text{dimension:} \left[\begin{array}{ll} \text{x:} & 100 \\ \text{y:} & 100 \end{array}\right] \end{array}\right] \\ \text{image\_boundary:} & \text{EVAL\_boundary\_function1} \end{array}\right]$$

Figure 10. Functional Processing Feature

3.1.4 Justification for a Feature Structure Based Representation

The feature structure is a very prevalent representation in the fields of natural language processing and logic programming [13]. In addition and seemingly unrelated, first-order logic terms are readily apparent in many AI areas of research. However, first order terms are often too limited for many AI applications. The feature structure representation, as a more powerful generalization of first-order terms, implements the additional flexibility required beyond first-order terms by most AI research fields, specifically computer vision. The properties of the feature structure; features labeled symbolically not by feature position, no fixed arity of terms required, and the demoted functor, make any type of information desired in the computer vision system representable within the feature structure. The ability to represent any type of computer vision information using a feature structure is the primary driving force of the feature structure as the bridge between the inherently numerical extracted image parameters and the level of abstract semantic information required to perform scene interpretation. The feature structure representation is a declarative representation of knowledge that also provides benefits in the areas of system flexibility, modifiability, and learnability [20]. Finally, many feature structure extensions that have not been taken into account in this work allow the feature structure representation to be extendible to a more structured programming framework for future computer vision system applications. These feature structure extensions include typed feature structures and type inheritance hierarchies for the feature structure representation [21, 19].

3.2 Feature Structures Containing Image/Scene Data

Image data, primitive features extracted directly from the image, complex models of the possible objects, as well as the grammar rules for combination of the feature structures may all be expressed as feature structures. Geometrical and viewpoint dependent, as well as semantic, information may be included in the building blocks (atomic feature structures) for the object model representations. The representations of different data may utilize different aspects of the feature structure such as a disjunctive or range-valued features. The implementation may also take advantage of the coreferencing aspect of feature structures to force equality constraints required by a computer vision model. In addition, because a feature value may be atomic or complex, the ability to build a feature value using another feature structure exists and is the most often used case with the feature structure representation.

3.3 Rules as Feature Structures

Rules in a unification grammar define how feature structures can be manipulated to build new feature structures. A key aspect of unification grammars is that a rule can be represented as a feature structure [10]. Application of a rule in the computer vision system proceeds by attempting to unify the feature structure representation of the rule with the current feature structure representation of system information. Rule application is permitted if the current feature structure does not violate any constraints represented as feature/value pairs of the rule. The unification process simultaneously enforces the constraints of the rule and builds the new feature structure that is specified by the rule. A sample feature structure representation of a rule is shown in Figure 11. This rule represents the knowledge that if there are a sufficient number of roads and houses identified in a low-level aerial photographic image, the system may add the scene information of a neighborhood in the image to the feature structure representation.

$$\left[\begin{array}{ll} \text{object:} & \left\{ \begin{array}{l} \left[\begin{array}{l}\text{name: house}\\ \text{number: (5, ..., 20)}\\ \text{viewpoint: top}\end{array}\right] \\ \left[\begin{array}{l}\text{name: road}\\ \text{number: (2, ..., 10)}\\ \text{viewpoint: top}\end{array}\right]\end{array}\right\} \\ \text{apriori:} & \left[\begin{array}{l}\text{source: low-level ariel}\\ \text{conditions: clear}\end{array}\right] \\ \text{image\_content: neighborhood} & \end{array}\right]$$

Figure 11. Feature Structure Representation of a Rule

There are many aspects of the feature structure representation that make rule representation very straightforward. The demoted functor aspect of the feature structure representation means there is no ordering of the information within the feature structure. As the system rules develop and expand, more information pertaining to a specific rule may need to be added or modified. Also, the consequences of a rule may need to be updated as the system progresses and expands. Because the information within the feature structure is order-independent, new information may simply be added to the original feature structure without modification, or original features may be modified without disturbing the remaining parts of the feature structure. The preconditions or consequents of a rule may be added in any order. There is no change required in the remainder of the system rules or the unification process to handle the addition of more or different rules. The order-independence of the feature structure and its manipulation process makes the computer vision system extremely flexible in terms of modifiability and extendability.

Another aspect of the feature structure representation is the separation of the variable representation into two distinct parts, the placeholder variable ([ ]) and the coreferencing. This makes the feature structure representation of rules very readable for the human system developer. The system developer or system modifier knows by sight if the feature was meant simply to hold a place for a future feature structure instantiation or was required to be completely equivalent to another specific feature (or feature structure). This coreferencing aspect also allows equality relationships within rule conditions to be readily implemented.

The final aspect that makes rule implementation desirable in the feature structure representation is really a by-product of the flexibility of the feature structure itself. Since all information within the computer vision system is representable as a feature structure, whether the information be rules, image data or features, or vision models, the representation is the same and, therefore, the manipulation process is the same. This means there is only one consistent implementation across the system to become familiar with and only one coherent scheme in the system to modify for future work. The desirability is high for an easily extendable computer vision system that can grow in the future and not become quickly obsolete because the representation cannot be modified. The feature structure representation handles this case of future expansion.

3.4 The Lattice of Feature Structures

It is common for feature structures to be viewed as directed acyclic graphs (DAG) and the process of manipulation is applied to the graph representation. In this work, however, the unification-based grammar formalism of feature structures was viewed from a somewhat different perspective. Instead of viewing the feature structures that compose the grammar as DAGs, they were seen as components within a lattice that is partially ordered by the subsumption relation. The use of the many types of feature structure information within this lattice structure allows for more efficient hypothesis generation and object classification while using the basic operation of unification to drive the classification process. The feature structure representation of object models and their components within a partially ordered lattice allows the incorporation of additional ideas, such as abstraction and hierarchical representation, to further enhance vision system flexibility.

In this research, the lattice of partially ordered feature structures is the most important structuring element and visualization tool within the computer vision system. There exist several issues that provide the theoretical understanding of how a computer vision system represented using feature structures (this includes the rules expressed as feature structures), partially ordered within a lattice, should function. These issues include the visualization of the lattice and the different aspects of the lattice representation itself, where does input data fit into the lattice structure, the structure of lattice information, how does abstraction/hierarchical processing fit into the lattice, and what types of data/features can be expressed in the lattice structure. These issues will be treated in turn in the following discussion.

3.4.1 Lattice Visualization

Conceptually, there are three lattices of concern within the computer vision system addressed here: the infinite lattice, the enumerated lattice, and the search lattice.

3.4.1.1 The Infinite Lattice

The infinite lattice is the representation that would exist if all possible models and elements that could be modeled within a physical world were present in the computer vision system. However, there will always exist more elements than can physically be represented within a computer vision system and, therefore, this lattice is impossible to realize. The infinite lattice is the conceptual whole from which the subset of elements to be modeled in a computer vision system will be taken. The infinite lattice is a many dimensional structure with the connections between feature structures within the lattice going into many directions, not just from top to bottom within a lattice as can be shown on the two dimensional page. The concept of the lattice going in the many possible directions determined by the feature structure connections is a crucial concept in the lattice visualization. These connections embody the power of the lattice structure that marks the possibility of moving to many different areas in a lattice through unification of feature structures and to explore any possible solution paths that may exist.

3.4.1.2 The Enumerated Lattice

The enumerated lattice is the lattice that, in theory, could be produced given the entire set of feature structure object models, feature structure rules, and feature structure inputs modelled in the current implementation of the computer vision system. In other words, this lattice consists of all feature structure unification paths that are possible given the set of objects and object components modeled as feature structures and the set of rules expressed as feature structures. This lattice is the complete enumeration of all unifications and feature structures possible within the computer vision system and is the complete subset of the infinite lattice that is physically modeled. While this lattice is never actually fully produced within the computer vision system, portions of the lattice are accessible for visualization purposes. Within sections of the enumerated lattice, the locality of information is readily apparent. As shown in Figure 12, portions of the enumerated lattice that can be identified as containing information about things such as houses, cars, or character information. These areas may

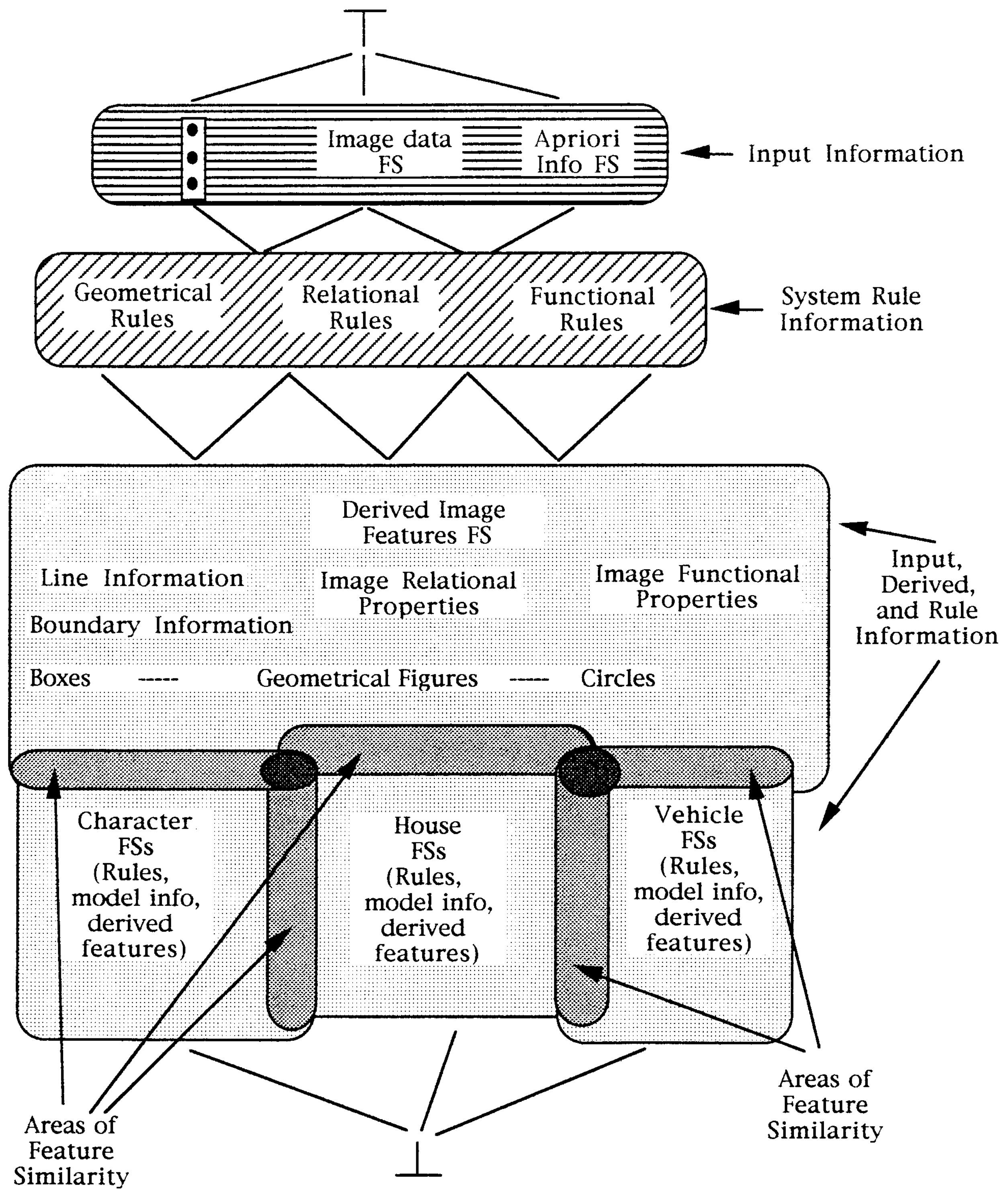

⊤
Image data FS
Apriori Info FS
Input Information
Geometrical Rules
Relational Rules
Functional Rules
System Rule Information
Derived Image Features FS
Line Information
Image Relational Properties
Image Functional Properties
Boundary Information
Boxes ----- Geometrical Figures ----- Circles
Input, Derived, and Rule Information
Character FSs (Rules, model info, derived features)
House FSs (Rules, model info, derived features)
Vehicle FSs (Rules, model info, derived features)
Areas of Feature Similarity
Areas of Feature Similarity
⊥

Figure 12. Locality of Information in a Lattice of Feature Structures

overlap where there are areas of common features, such as boxes or circles or the common structure of line information. The locality of information within the enumerated lattice is beneficial in partitioning the rules expressed as feature structures into sets for the purposes of unification and search. The feature structure portions within the enumerated lattice that distinguish the various regions of information are also used as an added control to the overall search through the enumerated lattice during the determination of image content.

3.4.1.3 The Search Lattice

The search lattice is the lattice that is actually produced in the search through the enumerated lattice during the determination of image content. This lattice contains only the feature structures and feature structure unification paths that are produced in this current search through the enumerated lattice to produce an image determination given the current image data. Not all feature structures and rules need to be used to determine what is contained in the current image. The search lattice is the subset of the enumerated lattice explored during the search for a possible image determination.

3.4.2 Input into the Lattice

All data, whether image data input or any other type of input information, comes into the lattice as a primitive feature structure, whether atomic or complex. In terms of lattice visualization of this input information, the data is input into the lattice at the top (T) of the lattice. This data, being expressed as a primitive feature structure with no additional information, does not arise as the product of the unification of other feature structures and rules within the lattice structure. Therefore, as this input information is not subsumed by other feature structures within the lattice, this input must be present at the top of the lattice (below the universal variable). At that point, this available input as a primitive feature structure may be used in possible unifications within the lattice to achieve the goal of the computer vision system.

3.4.3 Structure of Lattice Information

The lattice of feature structures is partially ordered by the subsumption relationship and this partial ordering determines the appearance of 2D or 3D information within the feature structures of the lattice. There are certain regions within the lattice that are strictly 2D in feature information and other regions that are strictly 3D in feature information. At some point in the lattice hierarchy, 3D features may either be derived from or combined with 2D features in the process of unification. When this occurs, both 2D and 3D features are combined into a single feature structure for use in achieving the goal of the computer vision system. Figure 13 shows a sample relationship of 2D and 3D feature information within the lattice structure.

The subsumption relationship within the lattice also allows for structuring for other types of information contained in the feature structure representation. Because the subsumption relationship only partially orders the lattice, only the feature structures that fall within the subsumption relationship are connected to one another through by a path. This partial ordering produces semi-isolated areas of information that are only sparsely connected to other areas feature structures by the subsumption relationship. Local areas of information relating to geometrical or structural commonalties,

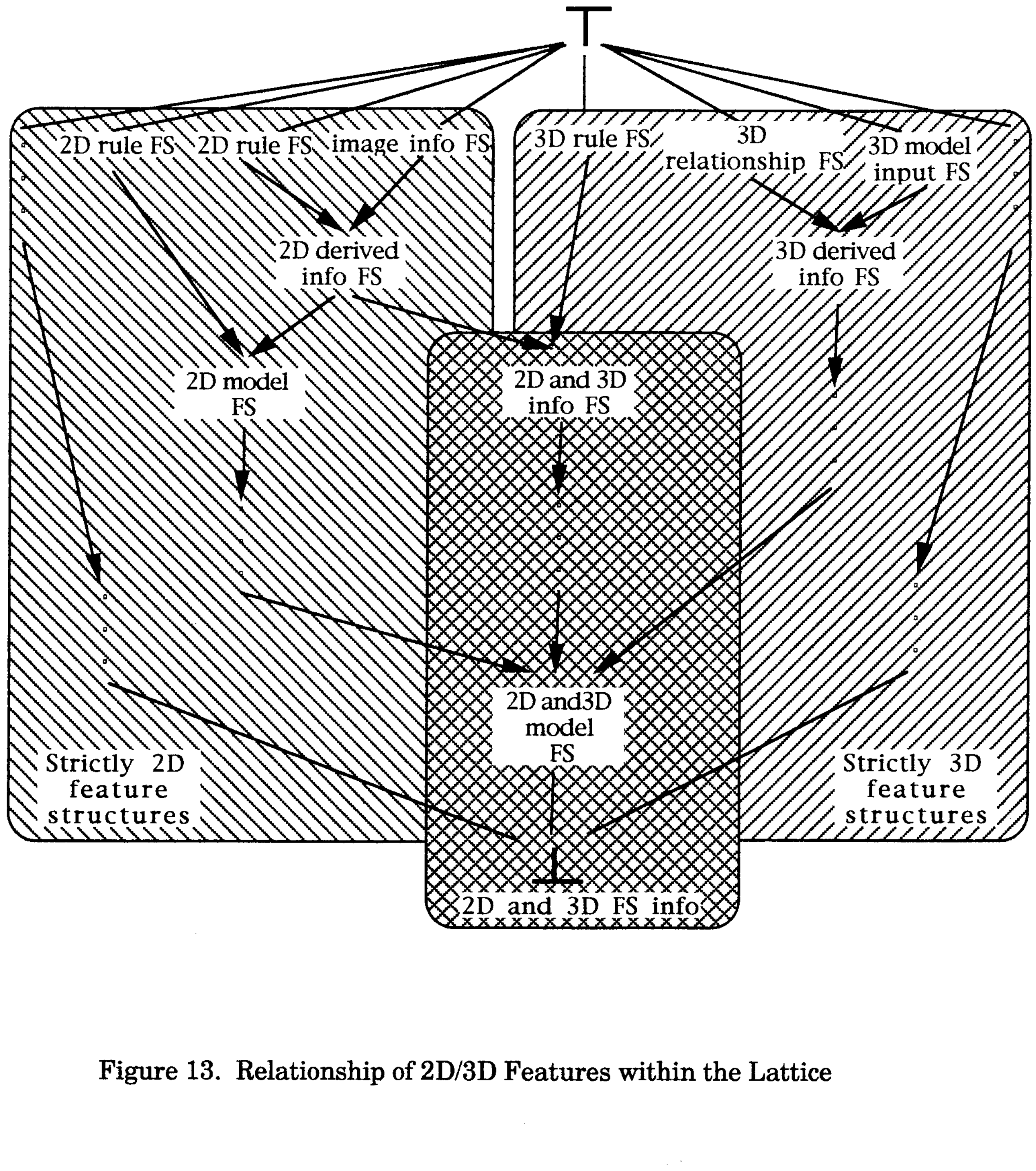


Figure 13. Relationship of 2D/3D Features within the Lattice

functional or semantic commonalties, or relational commonalties can be seen in the partially ordered lattice of feature structures. The local areas of feature structure information resemble the concentrations of 2D/3D feature structure information in particular regions of the lattice. An example of the regions of feature information can be illustrated using a category of `house objects'. These house objects have many characteristics, or features, in common and will have a portions of the subsumption based lattice in common. Therefore, at least some sections of the lattice can be identified as containing certain categories of objects and may then be used to concentrate the unification in the areas of interest to obtain a solution. Thus, hypothesis generation is possible using this type of lattice structuring information that is not actually information contained within a feature structure.

3.4.4 Abstraction/Hierarchical Processing within the Lattice

A hierarchical processing approach parallels the operation of the Human Visual System in that processing proceeds from a coarse descriptive level to levels of increasing refinement [17]. Within a computer vision system, a hierarchical processing ability is facilitated by a representation that allows for multiple resolutions, from a coarse to fine level of description, of the data to be analyzed and a structure for hierarchical processing. The incorporation of multiple resolution feature structures within the lattice as a method of feature value abstraction and the natural hierarchy imposed by the lattice structure extend the representational power possessed by viewing the feature structure in isolation and is appropriate for use in hierarchical processing.

Reasoning with multiple resolution feature structures within the lattice hierarchy work to bridge the gap from the extracted features with which the classification system must begin reasoning and the high level object models that are used for classification within a computer vision system. The abstract data representation allows for the extracted features to be expressed in the system directly as atomic feature structures that are combined into complex feature structures through rules expressed as feature structures and the unification process. The feature structures may be coarse or expressed at successively finer levels of detail as desired in the hierarchical representation. Likewise, feature abstraction may be accomplished using the initial pixel data expressed as feature structures and successively abstracting to the desired resolution level. These abstract feature values may then be used for reasoning at various resolution levels. Each level of feature structures (atomic through object model and at each resolution level) is expressed in the partially ordered lattice and can be accessed for purposes of reasoning about object classification and hypothesis generation.

3.4.5 Data Types within the Lattice

In the lattice of feature structures, any type of feature value, or data, may be present. The feature values may be information on geometric, relational, functional, or any other aspect of an object or image available. There is no restriction on the information that may be contained within the feature structure. The feature value may also be a processing rule or function that is evaluated as the unification using the feature structure is attempted.

An example of an image modeling technique that may be incorporated into a feature structure data representation is the aspect representation. An aspect representation, as previously discussed, is a geometric modeling technique that indicates the changes in the appearance of the object as a function of viewpoint [6]. The incorporation of this type of viewpoint-adjacency information into the feature structure representation makes it possible to reason about the change of object features with respect to viewpoint. This information can be inserted directly into the feature structure representation and then, using abilities of the feature structure representation, such as coreferencing, efficient traversal of the adjacency information through the lattice may be accomplished.

3.5 Feature Extraction

Feature extraction within the current computer vision system is done in levels. Because the image data is expressed as a feature structure, the vision system carries along the original data from which any additional features were derived. Therefore, if image explanation fails, the system may always generalize and reuse the original image data, boundary data, or line data (all expressed as feature structures) to arrive at possibly new or different feature determinations from the original image data. These new extracted features may then be used to generate another search path through the enumerated lattice and an image explanation may possibly be generated where before the image explanation failed.

The current system also makes the distinction between the data expressed as feature structures whose computation does not depend on knowledge of the scene content (e.g., edges or segmentation) and the derived features, also expressed as features structures, whose computation requires specific information about the scene (e.g., geometric figures or objects). The exercise of feature derivation is considered more high-level vision and uses rules expressed as feature structures to derive the additional feature information.

The current system ability to retain the original image data along with the ability to do feature extraction in levels addresses a major weakness in most high-level vision systems. In most high-level vision systems, the feature extraction portion operates (generating feature such as edge points, edges, and surfaces) autonomously from the rest of the vision system for interpretation by the high-level portion for image determination. This decoupling of the matching and feature extraction processes necessitates the computation of all potentially useful features. The management of these features during matching is potentially combinatorially explosive. In addition, the matching process distances itself too rapidly from the image data. An error in the extractions is usually irreversible [22].

The approach of the current system to express all information as feature structures; be it image data, boundary data, line information, object features or models, scene interpretation information, or rules for image feature derivations, provides a mechanism that proceeds gracefully from "low" to "high" level vision processes. The ability to reuse the "low" level vision information for additional "high" level feature extraction also provides a powerful reasoning source when image determination has failed. Thus, this abstract data representation based upon feature structures bridges the representational gap of computer vision systems between the image data, low-level processing representations of that data, and the high-level object and scene models of the real world.

CHAPTER 4
MANIPULATION TECHNIQUE

The manipulation technique for a computer vision system encompasses not only the actual manipulation approach used directly on the data structure representation of the vision system information, but also the control of that manipulation process in the search through the system information for an appropriate *image explanation.* Image explanation is a term that will be used to describe the computer vision solution to what is identified using the input image. In this chapter, the unification mechanism, its alterations and modifications, and its testing will first be discussed. The issues of searching through the lattice of feature structures, a process of generalization within the lattice of feature structures, and the system control of these processes is also discussed. Finally, the ideas on efficiency and scaling of this system process is addressed.

4.1 Unification as a Manipulation Process

Because everything related to this approach to model-based computer vision is based upon feature structures, a manipulation technique applicable to feature structures is necessary. The single operation used to manipulate feature structures is unification. Most systems of any type employing feature structures as the data representation technique (computational linguistics, logic system, computer vision) view the feature structures as isolated directed acyclic graphs and the process of unification is applied to the graph representation. In this work, however, the unification-based grammar formalism of feature structures is viewed from a different perspective. Instead of viewing the feature structures that compose the grammar as DAGs, the feature structures were seen as components within a lattice that is partially ordered by the subsumption relation. As stated previously, unification of feature structures can be defined by this partial ordering relation subsumption. The partial ordering imposed by subsumption provides a framework for characterizing unification [13] and the lattice structure for the visualization of the subsumption relationship between the feature structures contained in the computer vision system. The use of the many types of feature structure information within this lattice structure allows for efficient hypothesis generation and image content determination while using the basic operation of unification to drive the data manipulation process.

The addition of feature structure types to allow for representational flexibility and, more importantly, the incorporation of uncertainty reasoning into the feature structure representation dictate a change in the method of unification. Changing the notion of the feature structure represented as a DAG to viewing the feature structures within a lattice relationship also warranted a change in the method of unification. Because the DAG representation of the feature structures is not used in this work, the graph node unification algorithm is not used as it is for the DAG-represented feature structures. A method to match the list representation of the feature structure using the principle of greatest lower bound within the lattice was used instead. In addition, a robust treatment of uncertainty and increased flexibility of information representation in a computer vision system is possible using attributes of the feature structure representation such as disjunctive features, multi-valued and range-valued features, and functional processing feature values. These are complicating factors in any unification algorithm. In the current unification process, the unification algorithm does not require any differences in representation for the matching of feature structures to determine if they unify. However, the process of combining the information of the two original feature structures into the correct unifying feature structure is the portion that is modified from a simple conjunctive unification algorithm to allow features representations using disjunctive, multi-valued or range-valued, or functional processing feature structure attributes.

4.1.1 Conjunctive Unification

Conjunctive unification, known simply as unification unless another qualifier is stated, is an information combining procedure. Unification within the lattice is determined by the partial ordering of the subsumption relation. However, subsumption is only a partial order. Because of this, not every two feature structures are in a subsumption relation with each other. There are two reasons that feature structures may not have a subsumption relationship. One reason is that the feature structures have differing but compatible information (Figure 14). The second reason is the feature structures have conflicting information (Figure 15). The differences between these two cases is in the first case (Figure 14) there exists a more specific feature structure that is subsumed by both feature structures (Figure 16). However, in the second case (Figure 15), no such feature structure exists that is subsumed by both feature structures. The notion of combining the information from two feature structures, each of which may contain only partial information, to obtain a feature structure that includes all the information of both, which still may be only a partially defined structure, is

$$\left[\begin{array}{ll}\text{has\_image:} & \left[\begin{array}{ll}\text{input:} & \text{yes}\\ \text{dimension:} & \left[\begin{array}{ll}\text{x:} & 100\\ \text{y:} & 100\end{array}\right]\end{array}\right]\\ \text{apriori:} & \left[\begin{array}{ll}\text{source:} & \text{low-level ariel}\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{ll}\text{has\_image:} & \left[\begin{array}{ll}\text{input:} & \text{yes}\\ \text{dimension:} & \left[\begin{array}{ll}\text{x:} & 100\\ \text{y:} & 100\end{array}\right]\end{array}\right]\\ \text{apriori:} & \left[\begin{array}{ll}\text{region:} & \text{southeast\_us}\end{array}\right]\end{array}\right]$$

Figure 14. Feature Structures with Compatible Information

$$\left[\begin{array}{ll}\text{has\_image:} & \left[\begin{array}{ll}\text{input:} & \text{yes}\\ \text{dimension:} & \left[\begin{array}{ll}\text{x:} & 100\\ \text{y:} & 100\end{array}\right]\end{array}\right]\\ \text{apriori:} & \left[\begin{array}{ll}\text{source:} & \text{low-level ariel}\end{array}\right]\end{array}\right]$$

$$\left[\begin{array}{ll}\text{has\_image:} & \left[\begin{array}{ll}\text{input:} & \text{yes}\\ \text{dimension:} & \left[\begin{array}{ll}\text{x:} & 100\\ \text{y:} & 100\end{array}\right]\end{array}\right]\\ \text{apriori:} & \left[\begin{array}{ll}\text{source:} & \text{ground\_photo}\end{array}\right]\end{array}\right]$$

Figure 15. Feature Structures with Conflicting Information

the central idea of unification. In general, the interest is in the most general feature structure that contains all the information from the unified feature structures with no additional information. In formal terms, unification of two feature structures *D'* and *D"* is the most general feature structure *D*, such that *D'* subsumes *D* and *D"* subsumes *D* [14]. This most general feature structure *D* is also known as the greatest lower bound in the subsumption based lattice framework. In the second case (Figure 15), the feature structures contained conflicting information and there is no most general feature structure *D* that is subsumed by both feature structures. In this instance, unification of the two feature structures is said to *fail.*

$$\begin{bmatrix} \text{has\_image:} & \begin{bmatrix} \text{input:} & \text{yes} \\ \text{dimension:} & \begin{bmatrix} \text{x:} & 100 \\ \text{y:} & 100 \end{bmatrix} \end{bmatrix} \\ \text{apriori:} & \begin{bmatrix} \text{source:} & \text{low-level ariel} \\ \text{region:} & \text{southeast\_us} \end{bmatrix} \end{bmatrix}$$

Figure 16. Feature Structure Subsumed by Figure 14 Feature Structures

The following examples illustrate the notion of unification as an information-combining technique. Figure 17 shows unification is monotonic and is order-independent. Unification adds information only and no information is lost or subtracted when feature structures unify. It also makes no difference in which order the feature structures or features within the feature structures are unified. Figure 18 shows unification is idempotent. Figure 19 illustrates the variable as the unification identity element.

$$\begin{bmatrix}\text{object: house}\end{bmatrix} \sqcup \begin{bmatrix}\text{viewpoint: front}\end{bmatrix} \rightarrow \begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix}$$

Figure 17. Unification Is Monotonic

$$\begin{bmatrix}\text{object: house}\end{bmatrix} \sqcup \begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix} \rightarrow \begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix}$$

Figure 18. Unification Is Idempotent

$$\begin{bmatrix}\;\end{bmatrix} \sqcup \begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix} \rightarrow \begin{bmatrix}\text{object: house}\\ \text{viewpoint: front}\end{bmatrix}$$

Figure 19. Variables Are Unification Identity Elements

Figure 20 illustrates unification is different depending on whether the values are similar or identical. Figure 20 is crucial in showing the importance of reentrancy, or coreferencing, in unification. In Figure 20(a), the information is combined and only one of the features in the unifying feature structure is given additional information. In Figure 20(b), the features are coreferenced in the first feature structure and must be identical when combining the two feature structures. In this method, information about both features in the unifying feature structure is concluded from the single piece of information from the second feature structure.

Unification depends upon matching. The matching takes place between the two feature structures to be unified. The information in the feature structures must be matched to make sure the information contained in the feature structures is compatible before the combination of the feature structure information takes place. If the matching process determines that the information contained in the feature structures is not compatible, the feature structures cannot unify. Within the lattice framework, two feature structures that contain information that is compatible and will, therefore unify, are connected in the lattice by a greatest lower bound. On the opposite side, two feature structures containing information that is not compatible do not have a greatest lower bound feature structure that is subsumed by the two feature structures. The greatest lower bound of these two feature structures is the bottom of the lattice, or inconsistency, and unification fails. The process of determining if the greatest lower bound of the two feature structures is a feature structure or inconsistency is a matching process of the information contained in the two feature structures.

$$\begin{bmatrix} \text{component:} \begin{bmatrix}\text{subpart: window}\end{bmatrix} \\ \text{relation:} \begin{bmatrix}\text{component:}\begin{bmatrix}\text{subpart: window}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$\sqcup \begin{bmatrix}\text{relation:}\begin{bmatrix}\text{component:}\begin{bmatrix}\text{connects: on\_house}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{component:} \begin{bmatrix}\text{subpart: window}\end{bmatrix} \\ \text{relation:} \begin{bmatrix}\text{component:}\begin{bmatrix}\text{subpart: window} \\ \text{connects: on\_house}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

(a) Similar Values

$$\begin{bmatrix} \text{component: } \boxed{1} \begin{bmatrix}\text{subpart: window}\end{bmatrix} \\ \text{relation:} \begin{bmatrix}\text{component: } \boxed{1}\end{bmatrix}\end{bmatrix}$$

$$\sqcup \begin{bmatrix}\text{relation:}\begin{bmatrix}\text{component:}\begin{bmatrix}\text{connects: on\_house}\end{bmatrix}\end{bmatrix}\end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{component: } \boxed{1} \begin{bmatrix}\text{subpart: window} \\ \text{connects: on\_house}\end{bmatrix} \\ \text{relation:} \begin{bmatrix}\text{component: } \boxed{1}\end{bmatrix}\end{bmatrix}$$

(b) Identical Values

Figure 20. Unification Depends on Whether Values Are Similar or Identical

4.1.2 Disjunctive Unification

Disjunctive unification, an addition to the "regular" conjunctive unification case, is a method to extend the expressiveness of the feature structure based representation by the use of disjunctive features. Disjunctive unification is interpreted to mean that only one of the set of feature structures need be unified with, or, alternately, that a feature's value may be any one of the structures inside the braces. This is indicated by enclosing a set of (normal, conjunctive) feature structures within braces as in Figures 8 and 21. This type of general disjunction can be differentiated from so-called value disjunction, in which a feature is given a disjunctive specification as its value (Figure 21). Disjunction leads to a more powerful and expressive feature structure representation in which to model easily and effectively image data and features that were awkward to model using conjunctive feature structures only.

Changes to the usual straightforward information combining procedures are needed to implement the idea of disjunctive unification within the partially ordered lattice of feature structures. Although disjunctive feature structures are of great utility, their manipulation is difficult [13]. The integration of disjunction into feature system representations has also be shown to impact system performance, both theoretically by complexity results and practically by dramatically slowing down the average performance of implemented systems that supported disjunction [19]. In the feature structure representation, disjunction was a necessary addition and the performance of the system due to its implementation was a system tradeoff. When performing disjunctive unification, the system must make certain that all possibilities are maintained since the feature's value need only be one of the values contained in the disjunction. The implementation of disjunctive unification involves cross multiplying the values of the two disjunctive features [13]. Figure 22 shows an example of disjunctive unification. In this example, there are four possible ways to combine the values of the "component" feature; three are successful, and one fails (because of the conflicting assignment to the "subpart" feature).

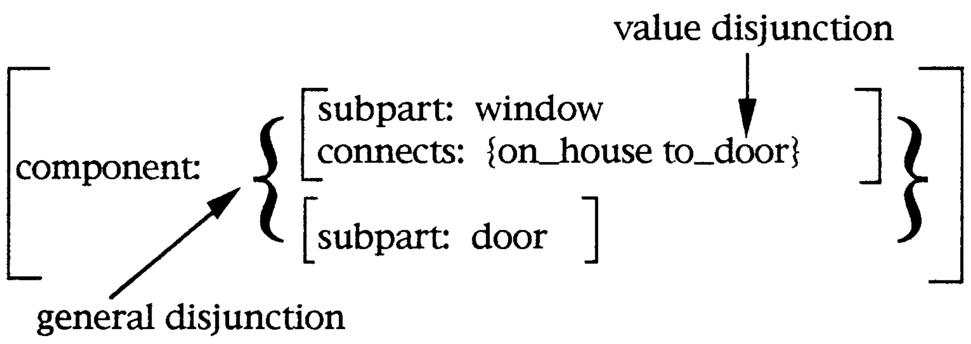


Figure 21. General and Feature Value Disjunction

4.1.3 Range-Valued and Multi-Valued Feature Unification

Like disjunctive features, the extension of the feature structure representation to include range-valued and multi-valued features dictated a change in the basic unification procedure. The features defined in terms of range-valued and multi-valued may take on more than one value. Unlike disjunctive features where only one of the values need be true, range-valued and multi-valued features require all of the values of the feature be true. The current work focused on the use of this feature structure extension to represent noisy image data, uncertainty in the features derived from the image, and in feature structure rules as a range value for the number of things that could possibly be contained in an image. As with disjunction, the use of

$$\begin{bmatrix} \text{component:} \left\{ \begin{matrix} \begin{bmatrix} \text{subpart: window} \\ \text{connects: on\_house} \end{bmatrix} \\ \begin{bmatrix} \text{subpart: door} \end{bmatrix} \end{matrix} \right\} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{component:} \left\{ \begin{matrix} \begin{bmatrix} \text{subpart: window} \\ \text{type: Pella} \end{bmatrix} \\ \begin{bmatrix} \text{viewpoint: front} \end{bmatrix} \end{matrix} \right\} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{component:} \left\{ \begin{matrix} \begin{bmatrix} \text{subpart: window} \\ \text{connects: on\_house} \\ \text{type: Pella} \end{bmatrix} \\ \begin{bmatrix} \text{subpart: window} \\ \text{connects: on\_house} \\ \text{viewpoint: front} \end{bmatrix} \\ \begin{bmatrix} \text{subpart: door} \\ \text{viewpoint: front} \end{bmatrix} \end{matrix} \right\} \end{bmatrix}$$

Figure 22. Disjunctive Unification

range-valued and multi-valued features leads to a more powerful and expressive feature structure representation in which to model easily and effectively image data and features that are vary awkward to model using conjunctive feature structures only.

The unification process is the same for both range-valued and multi-valued features and is shown by the examples in Figure 23. Because all of the values of a range-valued and multi-valued feature are interpreted to be true, the unification of two features with a multiple value is the most specific set of values that are true for both features, in other words, the intersection of the two sets (Figure 23(a)). In the case that one feature structure has a multiple value for a feature and the other feature structure contains only a single value, the single value is the most specific set (intersection of the two sets) and is the value for the unifying feature structure if the single value is contained in the multiple values of the other feature structure (Figure 23(b)).

4.1.4 Functional Processing Feature Unification

The last extension to the basic feature structure representation that required a change in the basic unification process is the addition of functional processing as the value of a feature. As stated previously, this type of feature value will be computed while the system is running instead of being a static value from the image data or derived by rule application during previous unifications. Because the unification process simply tries to match the two feature structures and then combine the feature information appropriately, unification does not require the feature values to be of any type, but only to not conflict. When functional processing is required during the matching process of unification, an indication that the value must be calculated at this point is inserted into the feature structure representation (demonstrated in

$$\begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } (5, \ldots, 20) \\ \text{viewpoint: top} \\ \text{color: brown} \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } (3, \ldots, 10) \\ \text{viewpoint: top} \end{bmatrix} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } (5, \ldots, 10) \\ \text{viewpoint: top} \\ \text{color: brown} \end{bmatrix} \end{bmatrix}$$

(a) Both Feature Structures Contain a Range-Valued Feature

$$\begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } (5, \ldots, 20) \\ \text{viewpoint: top} \\ \text{color: brown} \end{bmatrix} \end{bmatrix}$$

$$\sqcup \begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } 5 \\ \text{viewpoint: top} \end{bmatrix} \end{bmatrix}$$

$$\longrightarrow \begin{bmatrix} \text{object:} & \begin{bmatrix} \text{name: house} \\ \text{number: } 5 \\ \text{viewpoint: top} \\ \text{color: brown} \end{bmatrix} \end{bmatrix}$$

(b) One Feature Structure Contains a Range-Valued Feature

Figure 23. Unification of Range-Valued Features

Figure 10). When the matching procedure comes across this type of indication, the matching is suspended while the value is calculated by the specified function contained in the feature value and then this value is matched against the one contained in the other feature structure to unify. If both feature structures contain a functional processing value, both values are calculated using the appropriate functions (these may be the same or different functions) and then matched to determine unification success or failure. The value returned by the function then replaces the functional processing value in the current feature structure if unification is successful.

4.1.5 Unification Algorithm Testing

Because the unification algorithm is a computer manipulation technique, some degree of formal software testing is necessary. As such, the unification software was analyzed and tested according to a formal software testing technique.

4.1.5.1 Testing Criteria

A testing criteria was chosen for the unification software considering an appropriate relationship of the testing time and effort required to the total research effort. As the exhaustive testing of the unification software was not of primary concern in this research, the testing effort began by chosing an appropriate testing criteria for this research effort. The All Edges (Branch Coverage) testing criteria [23] was chosen for testing of the unification software. The All Edges testing criteria requires that at least one path containing each edge (branch) be exercised by a test case in the test suite. In other words, every conditional statement of the tested software must evaluate to both true and false and both of these conditions must be executed at least once over the suite of test cases. The All Edges testing criteria is not the most extensive testing criteria, nor is it the least [24]. However, this criteria was deemed appropriate for the level of testing desired in this research effort.

4.1.5.2 Unification Testing

Once the testing criteria was chosen, the remainder of the testing effort consisted of analyzing the unification software against the All Edges testing criteria, formulating test cases to satisfy this criteria and finally, showing the software met the chosen testing criteria by test suite execution. The unification software was analyzed and found to contain 15 conditional statements that required execution. Because of the nature of the LISP unification software, not all conditional statements led to only two cases to execute all branches within the software. The 15 conditional statements actually required 34 testing instances to cover all the branches. Eleven of these instances were unification failures and required separate test cases for each instance. The remaining cases were instances of unification success and could possibly be combined into fewer test cases. Some test cases were designed to exercise only one testing instance and some testing instances were easily combined into one test case. However for readability, no more than three branch execution instances were put into any one test case. Twelve additional test cases were designed for the remaining branch coverage testing of the unification software. In all, the test suite consisted of 23 test cases. The test case inputs executed using the unification software and satisfied the selected All Edges testing criteria. Considering this testing work and the many other executions of the software on image test data, the unification software performed correctly within the stated boundaries.

4.2 Unification and Search through the Lattice of Feature Structures

The goal of image interpretation in this computer vision system may be expressed as the simple unification of feature structures that produce a viable path through the enumerated lattice (a search lattice) to a possible solution determined by the input image. The unified feature structures may be atomic feature structures of image data or other input, rules expressed as feature structures for deriving additional feature information or feature processing, a complex feature structure that may contain an entire object or image model, or any level of feature structure in between the input and image models. The computer vision system problem can be solved, of a solution exists, by finding the unification path through the enumerated lattice to a world model, expressed as a feature structure, which will successfully unify with the image input. Examples of the simplified initial unifications required in the system and a simplified final unification within the lattice to converge upon a solution are show in Figures 24 and 25 respectively.

4.2.1 The Search Problem

Almost all AI programs can be said to be doing some form of problem-solving whether it be interpreting a visual scene, parsing a sentence, or planning a sequence of robot actions. Search is one of the central issues in problem-solving systems [25]. It becomes so whenever the system, through lack of knowledge, is faced with a choice form a number of alternatives, where each choice leads to the need to make further choices, and so on until the problem is solved.

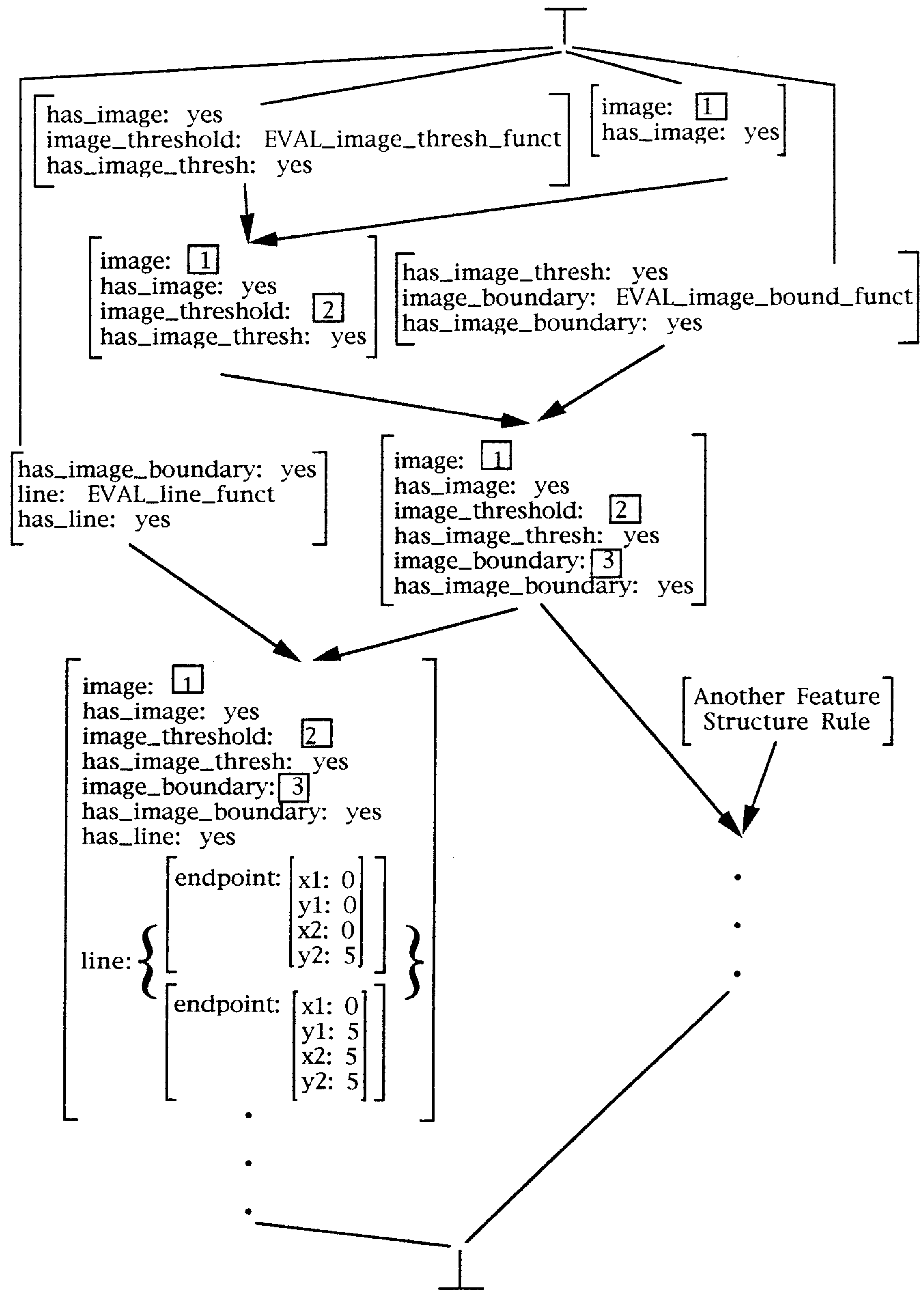


Figure 24. Initial Unifications of Input Feature Structures

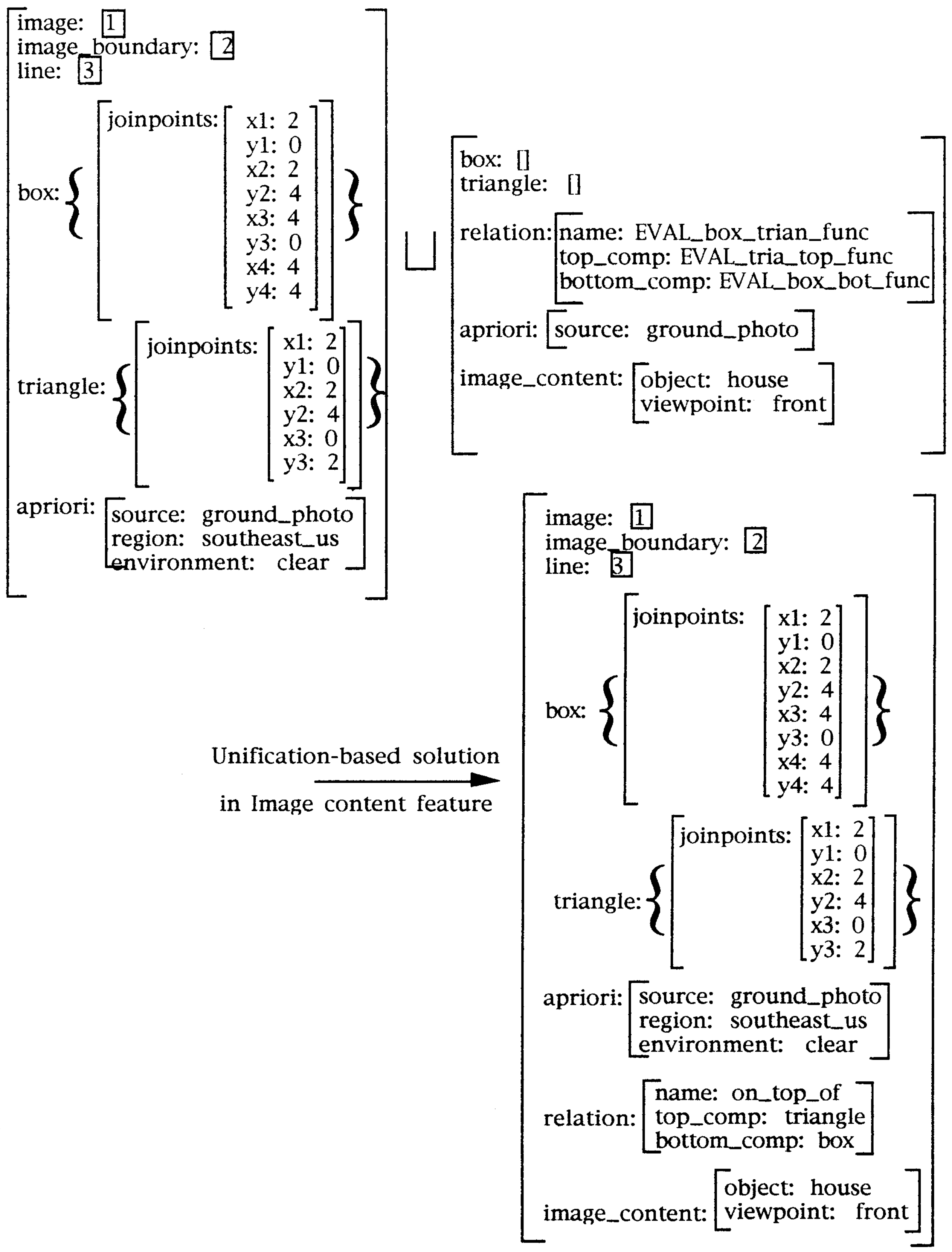


image: 1
image_boundary: 2
line: 3
box: joinpoints: x1: 2 y1: 0 x2: 2 y2: 4 x3: 4 y3: 0 x4: 4 y4: 4
triangle: joinpoints: x1: 2 y1: 0 x2: 2 y2: 4 x3: 0 y3: 2
apriori: source: ground_photo region: southeast_us environment: clear
box: []
triangle: []
relation: name: EVAL_box_trian_func top_comp: EVAL_tria_top_func bottom_comp: EVAL_box_bot_func
apriori: source: ground_photo
image_content: object: house viewpoint: front
Unification-based solution
in Image content feature
image: 1
image_boundary: 2
line: 3
box: joinpoints: x1: 2 y1: 0 x2: 2 y2: 4 x3: 4 y3: 0 x4: 4 y4: 4
triangle: joinpoints: x1: 2 y1: 0 x2: 2 y2: 4 x3: 0 y3: 2
apriori: source: ground_photo region: southeast_us environment: clear
relation: name: on_top_of top_comp: triangle bottom_comp: box
image_content: object: house viewpoint: front

Figure 25. Unification to Produce a Final Solution

4.2.1.1 The General Search Problem

A general search problem is characterized by the following [26, 27]: a set of system states that represent the problem state-space, a set of initial system states contained in the set of system states, a set of transformations (operators, rules) that can transform one state into another, and a set of goal states contained in the set of system states. Thus, problem solving is represented as finding and applying a sequence of operators that will transform an initial system state, through a sequence of intermediate states, into a system goal state.

In order to reasonably solve most AI problems, the computer system must avoid exhaustive searches of the state space. Two common methods of pruning the search are [26]: ordering the state-transformation operators such that the "best" applicable operator is always tried first; and defining a state-evaluation function that is used to rank intermediate states with respect to their "closeness" to a goal state (thus one can always choose to move to that next intermediate state that is closest to the desired goal state). Finding a "best" ordering of the operators and finding a useful "closeness" function are both difficult problems that are typically solved by the use of problem-specific heuristics. Within a computer system the critical decision making is concentrated in which transformation operator to apply and, hence, which state to move to next [26].

4.2.1.2 The Search Problem through the Lattice of Feature Structures

A current computer vision system using the representation of the lattice of feature structures exhibits the classical characteristics of the general search problem. The set of system states that represent the problem state-space are contained in the enumerated lattice. The initial system state is the image data and apriori information input into the system as a feature structure. The set of transformations (operators, rules) that can transform one state into another are the rules expressed as feature structures within the enumerated lattice. Finally, the set of goal states contained in the set of system states are the world models of the objects or scenes possibly contained in the input image. Therefore, problem solving in the current computer vision system is represented as finding and applying a sequence of feature structure rules that will transform an initial image data and apriori information (initial system state), through a sequence of feature structures contained in the enumerated lattice that add derived feature information to the original input data by unification (intermediate states), into a system goal state containing the explanation of the input image as a feature structure.

A possible solution path through the lattice, if it exists, may be visualized for small problems; however, with even modest problems, there are many possible search lattices and the decision of which features structures to unify in the production of a search lattice must be chosen from many possible applicable feature structures in the system. The critical problem solving step is to determine which grammar rules expressed as feature structures are appropriate at the given time and then decide which features structures may be unified with the chosen grammar rule. The possibility of many grammar rules and a lattice of feature structures at multiple levels of resolution and containing many types of feature information yields a search problem.

The choice of what feature structures are appropriate to unify and, hence, where to move next in the lattice is the critical decision in producing a vision solution, if it exists, in the current system. This choice of what to unify is the central issue of this computer vision search problem and must be addressed to avoid exhaustive search of the lattice. A possible solution to the question of what to unify may only be answered through system control.

4.2.2 System Control

An additional manipulation process required by this computer vision system is the control mechanism to decide which feature structures to unify as the guidance for search throughout the lattice of possible applicable feature structures. To simply pick the first rule and attempt unification with the first available feature structure could waste valuable processing time in possibly futile unification attempts. In the currently implemented vision system of only a few grammar rules (about 100) and a lattice of feature structures at multiple levels of resolution that contain many types of feature information, the search problem is readily apparent. An appropriate method is required to choose a grammar rule that is applicable in the current situation and then decide which feature structures may be unified with the chosen grammar rule.

At present, the control of search through the lattice is heuristic. These heuristics are implemented in the current system using meta-rules that reason about which grammar rules to unify at the current time in the vision system. These meta-rules guide the vision system's focus of attention. Different heuristics may be used in the search for a solution. The current system contains three different methods to guide the choice of what to unify next. The first and simplest is to pick the first meta-rule that is applicable. This technique is the easiest and quickest to make a decision on what to unify. With careful ordering of the grammar rules, this technique prunes the search space without needless control overhead. However, without careful rule placement, the unification path identified in this method may produce more failed unifications than one of the following techniques that require more control processing time to choose a unification path. The second method picks the meta-rule with the most specificity, or highest number of antecedents. The rational of this measure is that the meta-rule that matches the most conditions of the current feature structure will be closer to the correct unification path, if it exists. This method required more control processing time than simply picking the first applicable meta-rule, but also may produce less futile unifications than that method. The last method is to choose the meta-rule that puts the unification path closer to the goal state of containing the image explanation in the feature structure. This method is measured by determining a value for the grammar rules in the lattice. The higher the rule value, the closer the rule may be in the lattice to a possible goal state. The meta-rule consequent is a grammar rule with which to unify. Therefore, this method is implemented by choosing the applicable meta-rule with the highest value associated with its consequent. The drawback of this method is an incorrect goal state may be contained in the applicable meta-rule chosen and failed unifications are produced as a result.

The control system's meta-rules are production rules within a overall system control inference engine. The meta-rules contain information about the control of unification and what grammar rules may be applicable using the current system information. The applicability of a meta-rule is determined by matching the current system information contained in a feature structure against the meta-rule pre-conditions. If all meta-rule pre-conditions are met by the current system information the meta-rule is placed in a conflict set that contains all applicable meta-rules. A meta-rule is then chosen from the conflict set by the chosen control scheme and fired. The firing of a meta-rule by the system control inference engine produces controlled unifications of grammar rules and the current image information expressed as a feature structure.

Additional system control is implemented using partitioned grammar rules. The grammar rules are partitioned according to locality in the lattice and grouped into rulesets. This means the rulesets implement groupings of grammar rules with similar feature structure content and meaning. Using Figure 12 as an example, the rules would be partitioned into ruleset containing information about character identification, houses, and vehicles. Low-level image processing rules may also be divided into rulesets by the processing methods used to produce the desired image features. The rulesets are then considered as a group that can be unified with and not as individual grammar rules. The meta-rules then reason about which rulesets are applicable in the current situation. For the measure of closeness to a goal state, the entire ruleset is then given a value on closeness to producing a possible goal state instead of individual grammar rules.

When a meta-rule is fired, unification is attempted using each rule within the ruleset identified by the meta-rule as applicable. This strategy reduces the number of possibilities the control strategy must consider, but also has the possibility of more failed unifications. This strategy falls in between the extreme of trying to identify the "best" grammar rule to unify with and producing no failed unification attempts and the other extreme of blind unifications that produce many failed unification attempts. This strategy strikes a balance between the control process reasoning about which grammar rules to unify and the unification process that will actually produce the vision solution.

4.3 Generalization

There is also the option of generalization available in the search through the lattice. Generalization is often referred to as the dual of unification [13]. Recall that within the lattice framework, unification corresponds to finding the greatest lower bound (or meet) of two feature structures contained in the lattice. Alternately, generalization corresponds to finding the least upper bound (or join) of two feature structures contained in the lattice. The top of the lattice ($\top$), to which all pairs of terms can generalize, is also called the *universal term* or *universal variable.* Every feature structure contained in the lattice is an instance of this term [13].

In the current vision system, generalization is used to point to common aspects within an image and also point to other promising solution paths to explore. The generalization operators used in the current system were the dropping condition operator and the constants to variables operator [27]. The dropping condition operator relaxes conditions required by a grammar rule feature structure. The constants to variables operator relaxes required values contained in the feature structure models to variables. In this method, general aspects of a world model or feature structure grammar rule are retained, but some specifics of the model or rule may be relaxed to allow the unification search path through the lattice to proceed in other directions so that a viable solution may be determined.

Other uses of the implemented generalization technique could be accomplished in the current system. Its potential uses include learning, automatic lattice structuring and ruleset grouping, and creating meta-rules for the use of generalization in the lattice.

4.4 Unification Efficiency

The efficiency of systems based on the use of the feature structure representation is the subject of intense study and research in the linguistic community [19]. Many in that community have researched methods to implement unification efficiently. Most of these methods start with the feature structure represented as a DAG and reduce the overhead of DAG copying during unification by sharing data structures and delaying copying. While the implementation of feature structures and unification in the current system do not use DAGs, the principles embodied by lazy incremental copying [28] and structure sharing during unification were implemented. Where original algorithms ran in exponential time [13], ones employing the above principles ran in almost linear time [29].

A more serious efficiency problem is in the treatment of disjunctive feature structures. Kay [15] introduced disjunctive feature descriptions into his Functional Unification Grammar to make lexical descriptions compact and thus make them easy to understand. Disjunctive descriptions reduce the number of lexical descriptions. However, if disjunctive feature descriptions were expanded into disjunctive normal form (DNF), where the entire description can be interpreted as a set of conjunctive feature structures, the unification of the expanded feature structures would also take exponential time in the number of the original disjunctions [30]. There has also been much linguistic community research into this efficiency problem and unification methods for disjunctive feature descriptions that avoid DNF-expansion were shown to have an upper bound of $O(n^2)$ in the number of nodes contained in the input DAGs [29]. As with conjunctive unification, the current implementation of feature structures and unification in the do not use DAGs; however, DNF-expansion was avoided in the current unification algorithm for disjunctive features. The upper bound of $O(n^2)$ where n is the number of disjunctive feature values contained in the input feature structures remains applicable in the current unification portion to handle disjunctive features.

4.5 Scaling Factors

Along with the efficiency, the ability to scale a system model based upon the use of feature structures as a data representation and unification as the manipulation technique is of great concern to a practical computer vision system. Scaling issues include the following: (1) How easily can the vision system be expanded to include new world models, new rules, and innovations in other areas of computer vision? (2) What must be modified in the data representation or manipulation technique to include these new aspects? and (3) What impact does increasing the size of the system have on system efficiency?

As stated previously, an abstract representation based upon feature structures is the only one used in this computer vision system. Each piece of information within the vision system is modeled and manipulated in the same manner. The unification manipulation is order-independent. These factors make the addition of new models or rules a simple process. The development of the models and rules would be the actual work involved and the addition to the vision system would be as simple as typing the models or rules into the correct system files. In addition, feature structures make use of all techniques to represent computer vision system models. This means that developments in other areas of computer vision research can be directly applied in this representation. Research concerning geometric, relational, or functional modeling techniques may all be applied using the feature structure representation.

The second question relating to the modifications required in the current representation or manipulation technique should be apparent. No modifications are necessary to the unification technique or the feature structure representation for the addition of new information in the vision system. The changes required would only be in the area of system control. To utilize the new information, meta-rules would need to be developed concerning the appropriate use of the new information. In addition, the new information should be put into the appropriate ruleset for reasoning within the current system.

A possible issue relevant to system scaling is system performance. This issue may only be solved when the computer vision system is of a much larger size than the current implementation and system processing issues may be quantified. Current research in the linguistic community with regards to efficient execution of the unification algorithm could be applied at this point. Also, parallel implementations of the unification algorithm or parallel operation of many search paths through the feature structure lattice look highly promising and may help the system performance issue as the computer vision system is scaled to handle real world problems.

CHAPTER 5
EXPERIMENTAL RESULTS

Experiments were designed to illustrate the capabilities of the general feature structure representation and the manipulation and control mechanism as they relate to the solution development within the model-based computer vision system. The experiments were also designed to show the flexibility of the representational technique for the incorporation of new models and feature structures.

5.1 System Overview

The implemented computer vision system consisted of four parts: the vision system rules and models represented as feature structures, the unification software to manipulate the feature structures, a control system, and low-level image processing software to calculate functional processing feature values. The images were taken with a single vidicon camera and translated by a data translation frame grabber with a spatial resolution of 256x256 pixels and 64 gray levels. The image data was taken directly from the generic file representation of the frame grabber into feature structure representation. Once this representation of the image data was obtained, the remainder of system operation concerns the manipulation of feature structures. The low-level image processing software was accessed by functional processing features to perform operations such as thresholding, boundary detection, and segmentation. The data transfer between the low-level image processing software and the system feature structure representation was by data file transfers that were handcrafted to work within the current confines of the low-level software requirements and the feature structure representation requirements. Figure 26 shows a diagram of the computer vision components and their relationships. In Figure 26, the circled numbers indicate the flow of information and control through the system. A typical system simulation begins when the initial input image data, expressed as a feature structure, is input into the control system (shown by the circled 1). The control system uses this information and the meta-rules to choose a ruleset to attempt for unification. Control is then released to the unification software to attempt unification of the chosen ruleset and the current image information (shown by the circled 2). The unification software then tries to unify the chosen ruleset from the available rulesets in the vision system (shown by the circled 3) and the current image information. As necessary, the unification software may access the low-level image processing software to return a functional processing value when a functional processing feature is indicated a feature structure rule. After successful or failed unification, the unification software returns the current image information to the control system (shown by the circled 4) to process for another ruleset choice and unification cycle (circled numbers 2, 3, 4, and 5) or goal success. If goal success is indicated by the control system, the image identification as a feature structure is output to the system user (shown by the circled 6).

Because the rules expressed as feature structures are the main driving force for image identification in the experimental vision system, rule development for the experimental system was an important activity after the design and implementation of the unification and control systems. All reasoning processes for the derivation of image features from the original input image data and all world model information are contained in rules. The rules for the current system were developed using some previously developed

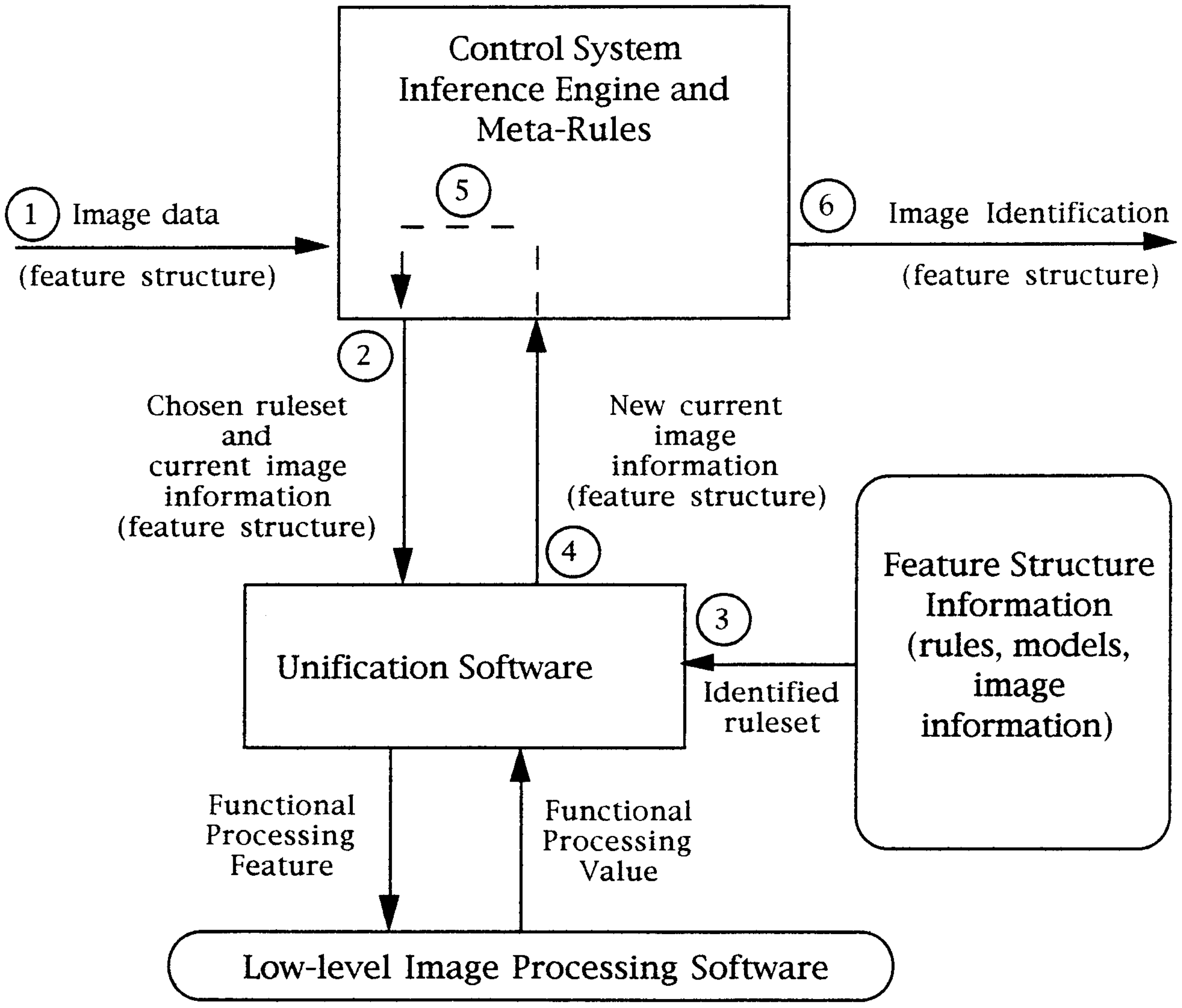


Figure 26. Experimental Computer Vision System

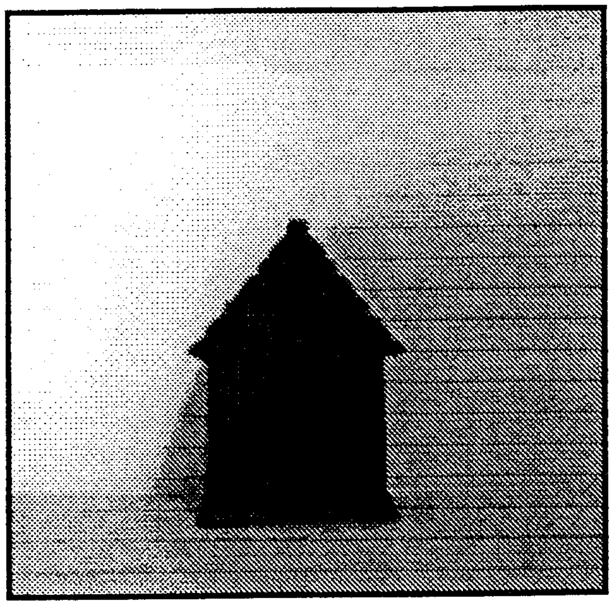
rule models and were expressed in the vision system as feature structures. Many of the rules in the experimental vision system were derived using Shaw's Picture Description Language [31]. Topa's Object Model Grammar [8] was used for reference in determining some necessary geometric features to derive using the geometric feature structure rules. In this manner, the experimental system also demonstrates that existing and available models may be expressed and manipulated using the abstract data representation presented in this work.

In addition, it should be noted that in all instances within the computer vision system, the entire enumerated lattice of feature structures containing rules, models, and image data/features is available for manipulation and control system use. The enumerated lattice structure is not expressly configured as a lattice structure within the vision system, but all feature structures in the system that would determine the enumerated lattice are available for processing and unification. This is stated to illustrate that inappropriate search lattices are available in the enumerated lattice in all experiments, not just the correct search lattices containing the appropriate rulesets are available for the reasoning process.

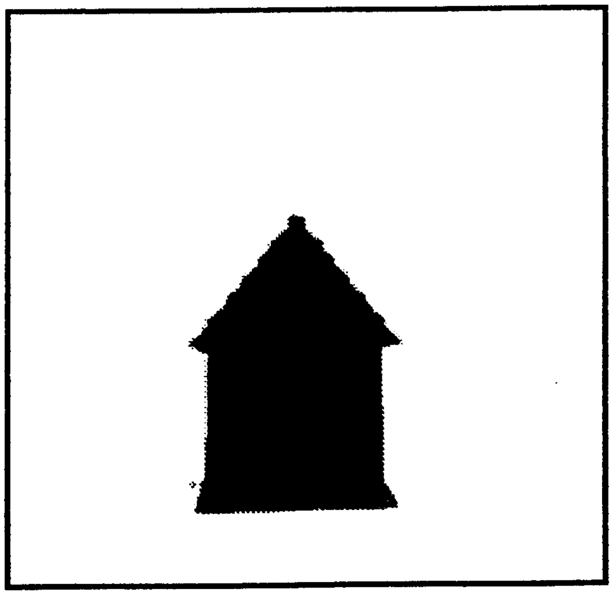
5.2 Experiment 1

The purpose of this experiment is to demonstrate the ability of the vision system to correctly determine image content when the object contained in the image is represented from different viewpoints. The same "house" was used as the experimental object in both parts of this experiment. The first image was taken from the front viewpoint of the house and the second image was taken from the side viewpoint of the house. The original image of the front viewpoint house, the image after thresholding, and the image after edge detection are shown in Figures 27(a), 27(b), 27(c) respectively. The original image of the side viewpoint house, the image after thresholding, and the image after edge detection are shown in Figures 28(a), 28(b), 28(c) respectively.

The two different viewpoints of the house were modelled as feature structures using geometrical and relational information. From the previous figures, the geometries of the same house from the two viewpoints are shown to be radically different. The initial image data, also represented as a feature structure, was used by the control system to determine an appropriate ruleset to unify with and try to derive additional information about the image content. Rules, expressed as feature structures, were unified with that allowed functional processing of the initial image data to occur and add low-level image processing data to the feature structure representation of the image content. This feature structure was then used by the control system to determine the next appropriate ruleset to unify with. The processes of unification with the chosen ruleset and the determination of the appropriate ruleset to unify with by the control system was repeated until the feature structure representation of the image data was able to unify with the feature structure model of the house. The system goal of image identification was obtained and the system ceased execution. The house from the two different viewpoints were correctly identified by the vision system using the above process. The results of the system simulation for the image descriptions of the house in Figures 27 and 28 are shown in Figures 29(a) and 29(b) respectively.

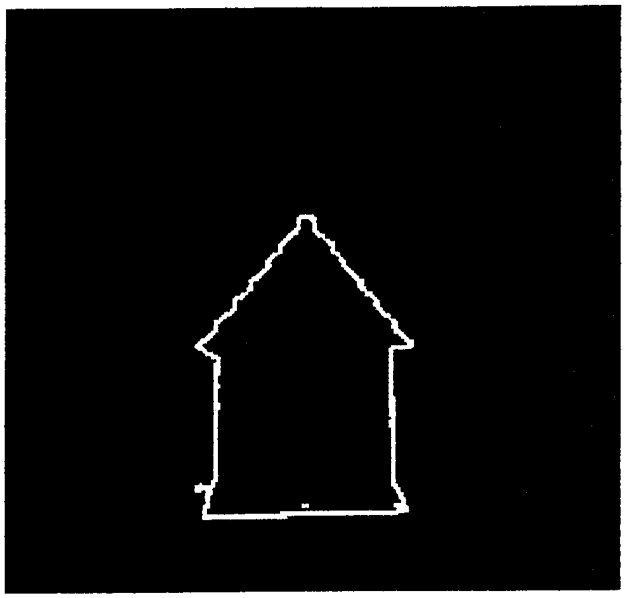


(a) Front Viewpoint House Original Image



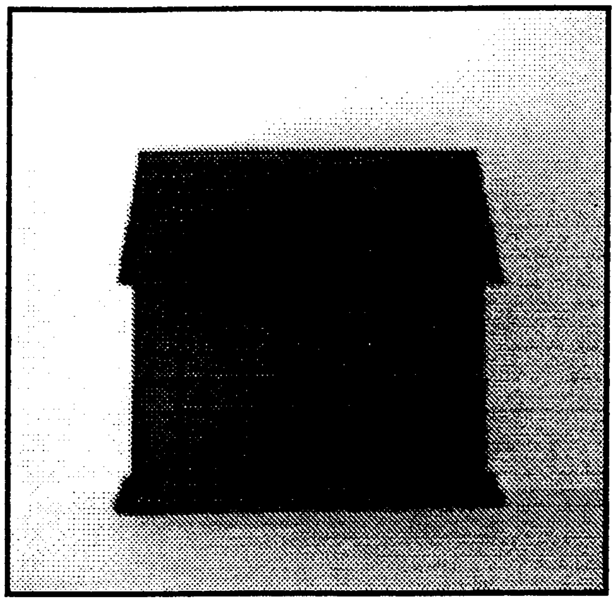
(b) Front Viewpoint House Image after Thresholding

Figure 27. Front Viewpoint House Experimental Images

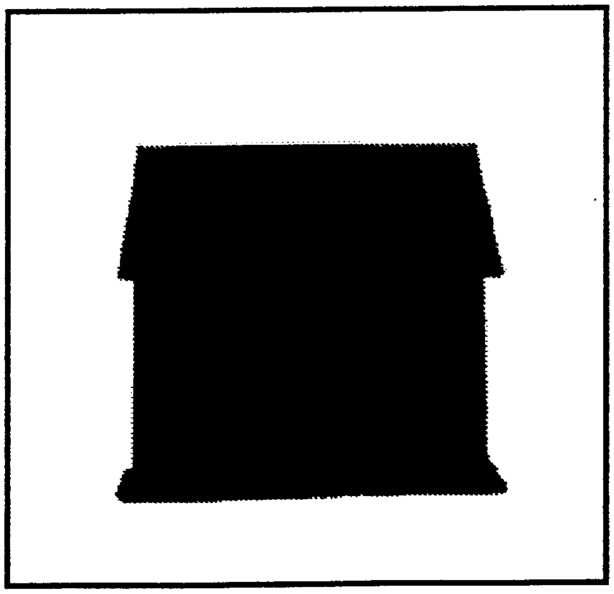


(c) Front Viewpoint House Image after Edge Detection

Figure 27. (Continued)

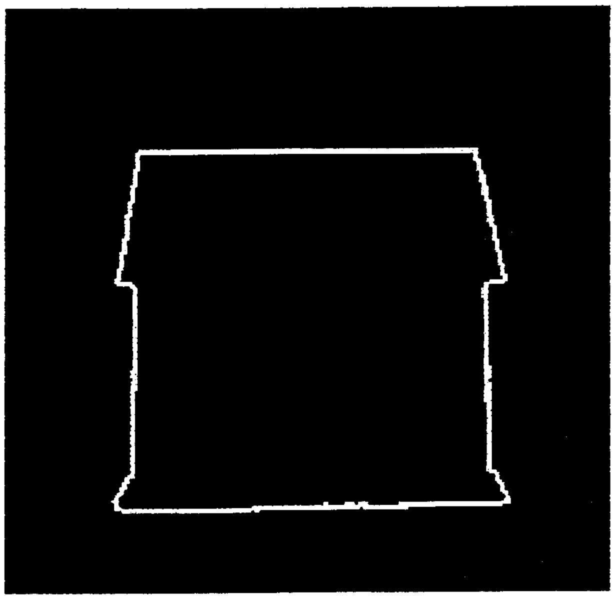


(a) Side Viewpoint House Original Image



(b) Side Viewpoint House Image after Thresholding

Figure 28. Side Viewpoint House Experimental Images



(c) Side Viewpoint House Image after Edge Detection

Figure 28. (Continued)

```
> (unify-lattice *image1* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*HOUSE-RULE-SET*)

(THE IMAGE EXPLANATION IS ->)

(IMAGE-CONTENT ((VIEWPOINT FRONT) (HAS-HOUSE YES)))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE1*)) ((@ 2) (VALUE *IMAGE1-THRES*))
((@ 3) (VALUE *IMAGE1-BOUND*)))
```

(a) Simulation Results Using Image of Figure 27

```
> (unify-lattice *image2* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*HOUSE-RULE-SET*)

(THE IMAGE EXPLANATION IS ->)

(IMAGE-CONTENT ((VIEWPOINT SIDE) (HAS-HOUSE YES)))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE2*)) ((@ 2) (VALUE *IMAGE2-THRES*))
((@ 3) (VALUE *IMAGE2-BOUND*)))
```

(b) Simulation Results Using Image of Figure 28

Figure 29. Simulation Results of Experiment 1

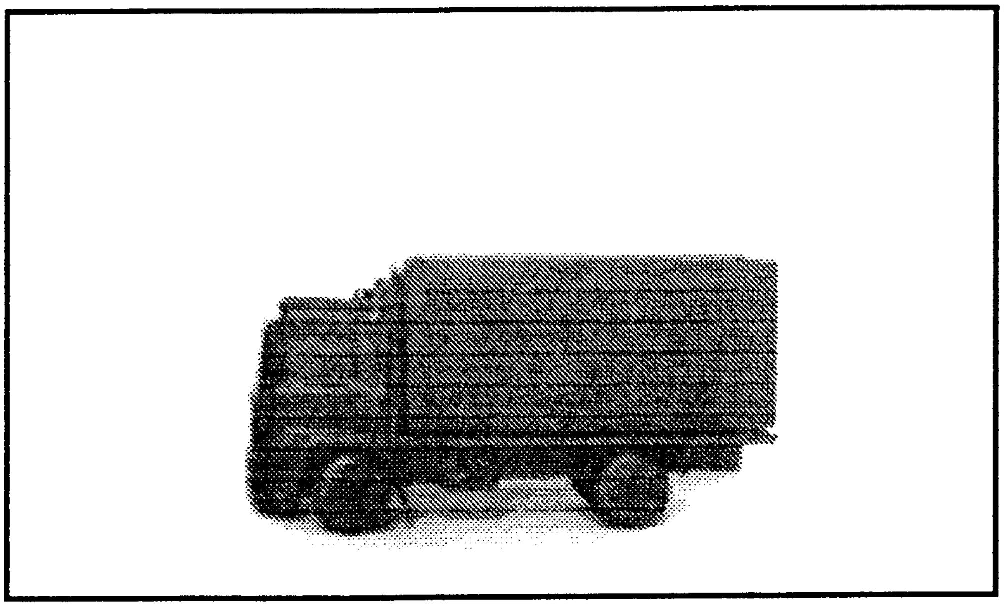
5.3 Experiment 2

The purpose of this experiment is to demonstrate the ability of the vision system to produce a result when the image content is not modeled in the vision system. The first part of this experiment consisted of a truck image that was modeled in the vision system. The original image of the truck , the image after thresholding, and the image after edge detection are shown in Figures 30(a), 30(b), 30(c) respectively. The second half of the experiment used the image of a car. The original image of the car, the image after thresholding, and the image after edge detection are shown in Figures 31(a), 31(b), 31(c) respectively.

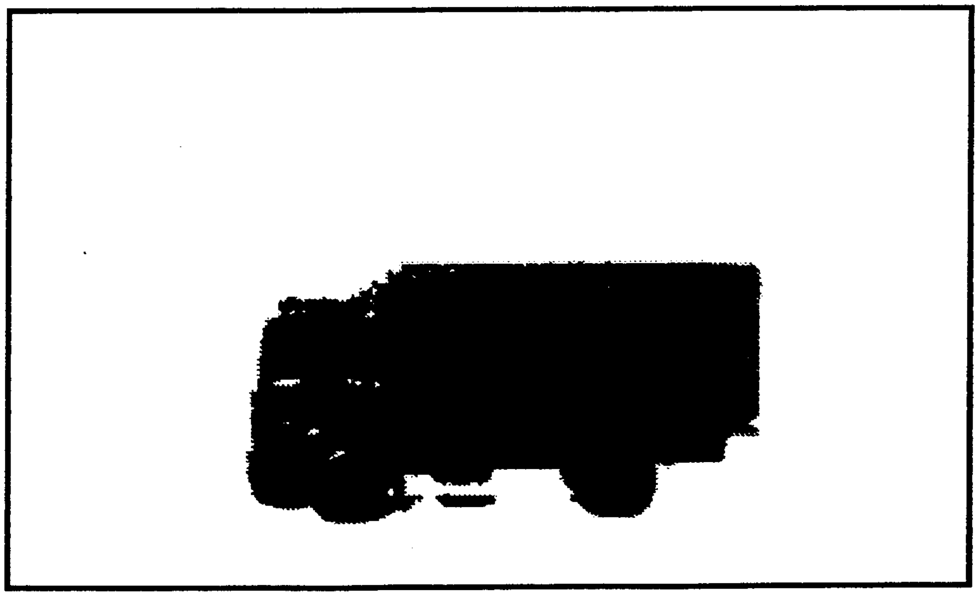
The truck was modeled using a feature structure representation similar to that of the house. From the previous figures, the geometries of the truck and car show similarities but are still significantly different. The differences of the car from a truck or a general vehicle were not modeled in the system and the features determined from the car image did not unify with the model present for a truck. The control and manipulation system used the initial image input to generate an appropriate search lattice from the different possibilities available within the vision system. The system then correctly presented the image determination of a general vehicle with the feature to label the specific type of vehicle in the model still undetermined. The simulation results using the images of Figures 30 and 31 are shown in Figures 32(a) and 32(b) respectively.

5.4 Experiment 3

Experiment 3 was designed to demonstrate the capabilities of the computer vision system modification capability by the incorporation of new models and feature structures to the existing system. The previous

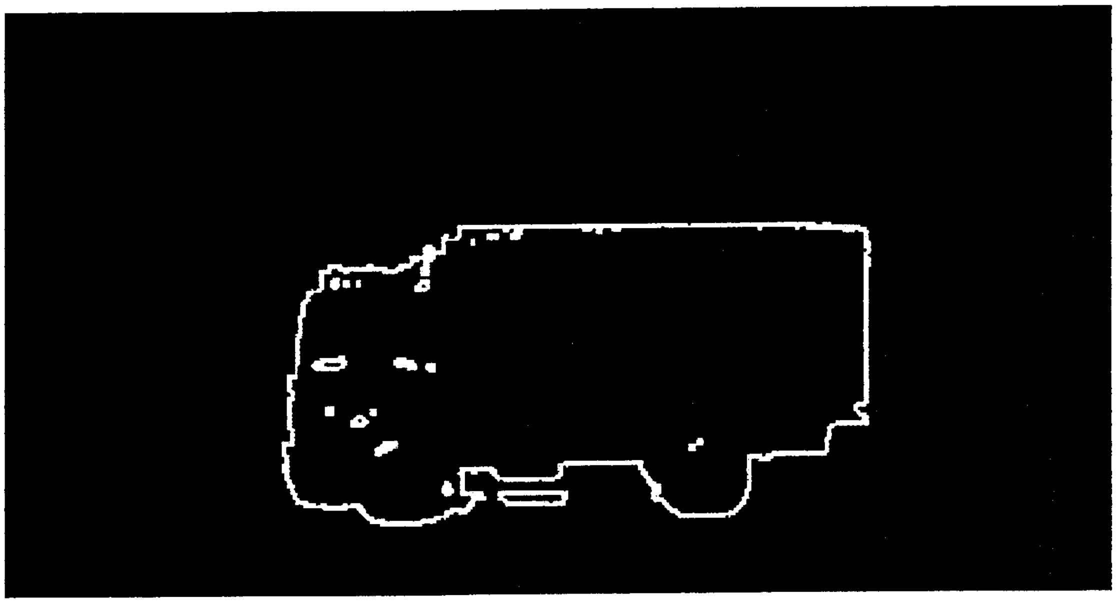


(a) Truck Original Image



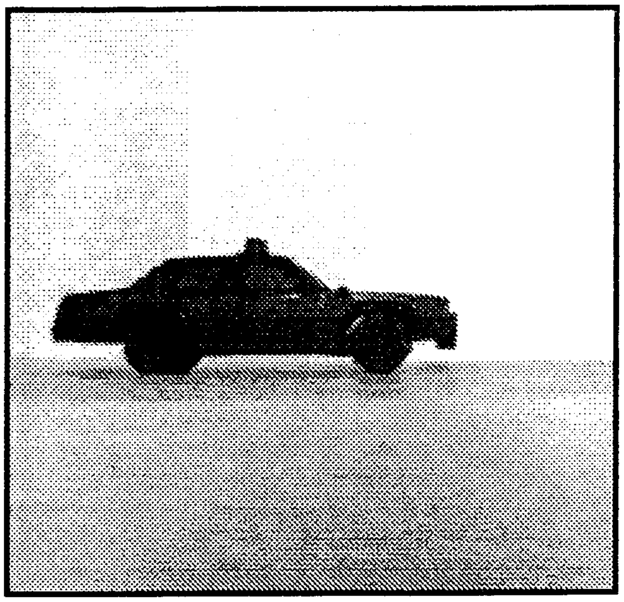
(b) Truck Image after Thresholding

Figure 30. Truck Experimental Images

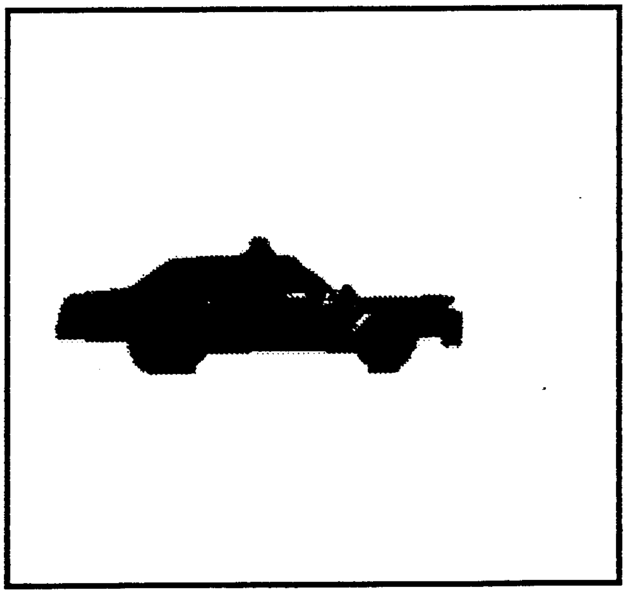


(c) Truck Image after Edge Detection

Figure 30. (Continued)

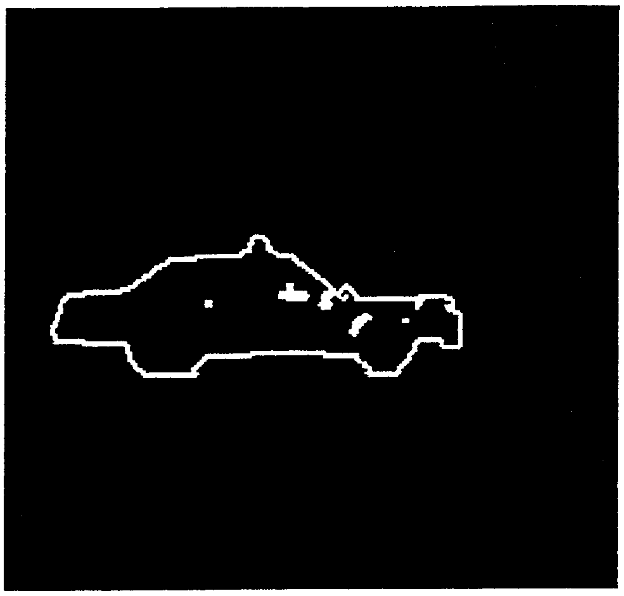


(a) Car Original Image



(b) Car Image after Thresholding

Figure 31. Car Experimental Images



(c) Car Image after Edge Detection

Figure 31. (Continued)

```
> (unify-lattice *image3* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*VEHICLE-RULE-SET*)

(THE IMAGE EXPLANATION IS ->)

(IMAGE-CONTENT ((VIEWPOINT SIDE) (HAS-VEHICLE YES)
(VEHICLE-TYPE TRUCK)))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE3*)) ((@ 2) (VALUE *IMAGE3-THRES*))
((@ 3) (VALUE *IMAGE3-BOUND*)))
```

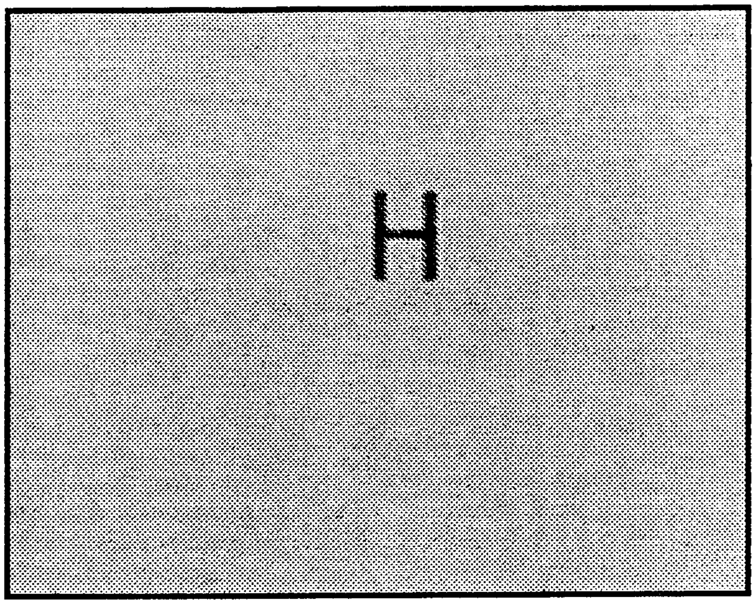
(a) Simulation Results Using Image of Figure 30

```
> (unify-lattice *image4* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*VEHICLE-RULE-SET*)

(THE IMAGE EXPLANATION IS ->)

(IMAGE-CONTENT ((VIEWPOINT SIDE) (HAS-VEHICLE YES)))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE4*)) ((@ 2) (VALUE *IMAGE4-THRES*))
((@ 3) (VALUE *IMAGE4-BOUND*)))
```

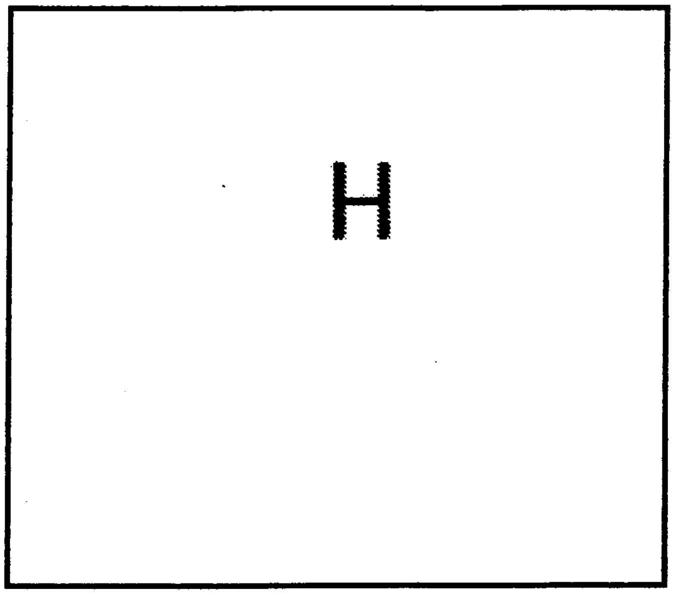
(b) Simulation Results Using Image of Figure 31

Figure 32. Simulation Results of Experiment 2

experiments were executed using an original set of rules and control system meta-rules. An image containing a block character H was then presented to the system for identification. The original image, the result of thresholding the image, and the result of edge detection applied to the image are shown in Figures 33(a), 33(b), and 33(c) respectively. The vision system correctly stated the presence of line segments and character boundaries. The vision system was not able to reason further due to the lack of more specific information on this type of image input. The current system was then modified by the addition of a ruleset that contained rules expressed as feature structures dealing with character identification. The addition of two meta-rules to the control system was all that was required to allow the system to begin reasoning using the new rulesets. With the addition of the new ruleset and meta-rules to deal with character images, the vision system then correctly used the new information in the appropriate manner to determine the block H character image content. The results of these two simulations are shown in Figures 34(a) and 34(b).



(a) Block H Character Original Image



(b) Block H Character Image after Thresholding

Figure 33. Block H Character Experimental Images

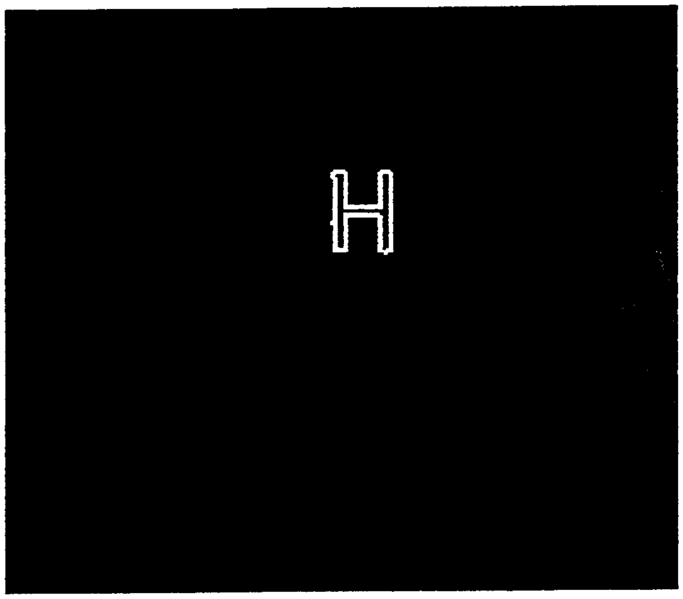

(c) Block H Character Image after Edge Detection

Figure 33. (Continued)

```
> (unify-lattice *image5* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*HOUSE-RULE-SET*)

(NO MORE APPLICABLE RULESETS)
```

```
(THE IMAGE EXPLANATION IS ->)

((HAS-IMAGE YES) (HAS-IMAGE-THRES YES)
(HAS-IMAGE-BOUND YES) (HAS-GEO YES)
(HAS-LINES YES))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE5*)) ((@ 2) (VALUE
*IMAGE5-THRES*)) ((@ 3) (VALUE
*IMAGE5-BOUND*)))
```

(a) Simulation Results Using Image of Figure 33 (Without Addition)

```
> (unify-lattice *image5* *meta-rules* 'first)
(USING RULESET ->)
(*IMAGE-INPUT-PROCESSING*)

(USING RULESET ->)
(*GEO-RULE-SET*)

(USING RULESET ->)
(*DESC-RULE-SET*)

(USING RULESET ->)
(*CHARA1-RULE-SET*)

(THE IMAGE EXPLANATION IS ->)

(IMAGE-CONTENT ((VIEWPOINT FRONT) (HAS-CHARACTER YES)
(CHARACTER-TYPE H)))

(THE VALUE OF THE BINDING LIST IS ->)
(((@ 1) (VALUE *IMAGE5*)) ((@ 2) (VALUE *IMAGE5-THRES*)) ((@
3) (VALUE *IMAGE5-BOUND*)))
```

(b) Simulation Results Using Image of Figure 33 (With Addition)

Figure 34. Simulation Results of Experiment 3

CHAPTER 6
APPROACHES TO PARALLEL IMPLEMENTATION

Some mention as to the possible benefits of parallel unification of DAG represented feature structures is present in Knight [13] and a parallel constraint satisfaction formulation dealing with disjunctive feature structures is presented by Hasida [32]. However, there exists no parallel treatment of the current unification within the lattice structure. One possible parallel unification algorithm is discussed along with a possible implementation platform. In addition, analysis of the issues related to unification in parallel of many search paths through the lattice of feature structures within a unification grammar framework is not present in the literature and was accomplished as part of this research work.

6.1 Parallel Unification

The process of parallel unification can be thought of as one way to build a fast unification engine. The benefits of parallel unification have been argued both positively and negatively by many in the linguistic community. It is this author's opinion that a parallel unification process may obtain positive results under the given circumstances. This section will first discuss the differences between a parallel unification approach for the current computer vision system and that of the standard implementation of parallel PROLOG [27] and how some of the problems with parallel PROLOG were avoided. Then an approach to parallel unification will be discussed along with the choice of an appropriate implementation platform.

6.1.1 Avoiding the Problems of Parallel PROLOG Unification

The major concern in Parallel PROLOG is with *argument dependence*, or shared variables, in the AND parallelism portion of Parallel PROLOG [27]. AND parallelism arises when two or more rule-antecedent clauses are to be verified independently in the parallel implementation. *Argument parallelism* refers to the process of unifying several predicates containing one or more variables in parallel. The shared variables may lead to binding conflicts. Therefore, the generation of independent sub-goals (or parallel processes) with shared variables in Parallel PROLOG is difficult. Approaches to handle the shared variable cases include removing all shared variables or conflict reconciliation using an additional unification step.

The unification of feature structures in the current computer vision system does not carry the argument dependence problem that is of major concern in Parallel PROLOG. Because the current system only compares two feature structures at a time for possible unification, the feature structures either unify or they do not. If one part of the feature structure unification process sets a variable and another portion of the feature structure needs the same variable to have a different value, these two feature structures are not unifiable and the unification process will fail. The current unification process is not exploring sub-goals that may possibly take on a different values that will eventually make the unification process succeed, as in PROLOG. In the unification of feature structures, only if both feature structures try to set the same feature to the same value or one feature structure sets the value and one feature structure is a variable will the unification process succeed. In this instance, it is of no consequence which event occurs first. Finally, shared variables do exist; however, in this work, they must either be set to the same value or unification fails.

The other major problem to be overcome in Parallel PROLOG is the issue concerning backtracking. Backtracking is the method that allows variables in the independent sub-goals to be reset again and again until a consistent binding for the shared variable is obtained or the data base is exhausted. Backtracking within the unification of independent sub-goals causes argument dependence to become such a problem. Backtracking of this nature is not allowed in the current unification of feature structures. The only aspect that might be considered backtracking is generalization of feature structures. Generalization occurs if a solution path in the lattice is unsuccessful in producing an image interpretation. Generalization of feature structures to previous a feature structure (or variant of a previous feature structure) is allowed and may generate a new search direction. At this point, the control system uses the new feature structure to explore another possible solution path in the lattice. With generalization of the current feature structure, the bindings upon the variables contained in the feature structure is also available for generalization. Any variable binding attempted during unification is not permanently altered until unification succeeds with the two feature structures. Therefore, all variable bindings not generalized out by the generalization of the current feature structure are valid to proceed with in the new possible solution path and do not need to be re-checked for consistency or altered in any reconciliation scheme.

6.1.2 A Parallel Unification Approach

The major process for parallel execution within unification of feature structures is the feature matching process. The matching of features for the unification of two feature structures determines if the two individual features contain conflicting or compatible information. If conflicting feature values exist in the two feature structures, unification fails. If the values are compatible, the feature information may be combined as necessary and the unification of the feature structures continues to the matching of the next feature within the feature structure. A scheme explored in this research is to match individual features from the two feature structures in parallel. There are two methods, one only a simplification of the other, that lend themselves directly to the parallel matching of features within feature structure unification.

A simplistic, parallel approach is to place the individual features of the current feature structure on different processors. The master processor in the parallel architecture contains a hash table that indicates on which processor in the parallel architecture each feature of the current feature structure is located. The master then sends the appropriate feature from the feature structure to unify to the appropriate processor for matching. The matching of the individual features of the feature structure to unify against the individual features of the current feature structure is accomplished in parallel. A processor reports back to the master the success or failure of the match and the result, if successful. Assuming the average number of features to match in a feature structure is n and the average time to match the feature against the current feature is t and the overhead to find the feature in the feature structure to match and subsequent unification success/failure reporting time is o, then the non-parallel time NPT to unify two feature structures is $NMP = nt + no$. Using the same assumptions as before for n, t, and o, the theoretical best case parallel time PT to unify two feature structures is $PT = t + no$. Given that the best case speedup for parallel unification over serial unification is $SPEEDUP = NPT \ / \ PT = \ (nt + no) \ / \ (t + no)$ and nt is large considering the number of features to match is large in comparison to *no*, the speedup may be given by $SPEEDUP = n$.

In reality, the speedup is less than this best case scenario; however, a practical speedup using an average case parallel matching scheme is possible. Degradation in the best case speedup is caused by various reasons. The overlap of the time to match features will not be complete or equal with each feature. This case does not take into account processor loading or the even distribution of features on the processors. In other words, all features of a feature structures are not created equal. One feature may contain a complex feature structure as its value while another feature may have only a simple numerical value. In addition, a feature may have a disjunctive value that takes considerably more time to match than the average conjunctive feature. The time to match these values in parallel to see if unification is possible is not equal, nor or the processors loaded equally. Also, in the more realistic case of a large number of features in the current feature structure to match against, each processor in a parallel processing scheme would have more that one feature from a feature structure for possible matching. The loading of processors with more than one feature happens for two reasons. First, the number of individual features in a fielded computer vision system feature structure would be much larger than the number of processors in the parallel architecture. Second, when each processor has more than one feature from the current feature structure, the processor is more likely to have a match with a feature in the feature structure to unify. In this manner, more processors are busy during unification and less computation power is idle using the parallel processing scheme. It is possible to use a control algorithm that is executed periodically that may compute a complexity measure using feature path length and the number of disjunctive features. This complexity measure would be used to evenly distribute the features over the processors. In this manner, more complex features that will take longer to match are placed on lightly loaded processors and relatively simple features that take less time to match individually will be placed on processors with other similarly less complex features. This scheme evens the time to match features and the theoretical speedup for this more realistic parallel scheme is n / n' where the dividing factor n' is the average number of features to match on a processor.

A single-instruction, multiple-data (SIMD) parallel machine would be an appropriate choice for the implementation of this type of parallel unification. Since all processor nodes are performing the matching operation, the same operation is being accomplished on each processor using the different features of the feature structure. Many processing nodes with the ability to store and execute the recursive matching procedure and also the memory to store at least one of the features of the current feature structure for matching against are required in the chosen SIMD machine. The actual number of processing nodes required is a function of the number of features contained in an average feature structure. Considering a fielded computer vision system where parallel unification may be required for efficiency, the number of features in the feature structure may be in the thousands and these would be distributed in groups of hundreds on the chosen number of processing nodes. As there are current machines that meet these processing requirements, any one of these parallel platforms that allow ease of implementation of the current unification matching scheme would be appropriate for implementation of this parallel unification algorithm.

6.2 Unifications in Parallel

The way to best improve the efficiency of a computer vision system relying upon the feature structure representation may not be parallel unification. Another method of parallel execution within the current computer vision system is to perform many unifications in parallel. This is not the concept of a fast unification engine as is the concept of parallel unification, but of the generation of many search lattices out of the possible enumerated lattice in parallel and determining the solution feasibility of the different search lattices in parallel. This means that a different search lattice would be explored by a separate processing node or set of processing nodes. Each alternate unification path through the lattice (search lattice) is generated independently with its own set of variable bindings. The unification paths in parallel, therefore, do not have the problem of argument dependence.

For a unification in parallel scheme, the master processor begins by executing the control system using the current image input data represented as a feature structure. The control system determines an ordered list of applicable rulesets at this point. The processing node then sends the current feature structure and a ruleset from the list to each connected node within a specified tree structure. This limits the path expansion available at each processing step and prevents the solution space from growing too rapidly. The next processing node in the tree then attempts unification of the current feature structure with the identified ruleset. If successful, the processing node repeats the same steps as its parent concerning the control strategy and sends its results to its connecting processors. Once a processing node has finished its current unification and control processes and sent its information on to the connecting nodes, it is again available for processing. In this parallel strategy, once a goal is identified in any of the processing paths by its use of the control system, processing is halted and the solution sent to the master and presented to the user. If interaction is available from a user, the presented solution may be accepted or rejected. If rejected, processing resumes at the saved nodes until an accepted solution is obtained. If unification fails at a processing node, that node signals it is available for processing and saves the unsuccessful path history with the master (so that it is not explored repetitively). If no solution is found, termination would be time limited and the current solution possibilities presented to the user ranked by specificity of the feature structure (a measure counting the number of features and length of feature paths). In this unification in parallel strategy, the master does one control system execution to start the search process and the remaining processing nodes do the rest. The master processor is waiting for a result from any possible processing node that has completed a unification and control system step.

To evaluate the speedup of the unifications in parallel approach a few assumptions are made. Assuming the average number of unifications in a search lattice is n, and the time required to unify two average feature structures is t, and the average number of search lattices that must be produced through unification to find a solution is l, then the non-parallel time NPT to find a solution to the content in the input image is $NPT = ntl$. Using the same assumptions as before for n, t, and l, the theoretical best case parallel time PT to find a computer vision solution for the input image is $PT = nt$. Given that the best case speedup for parallel unification over serial unification is $SPEEDUP = NPT \ / \ PT = ntl \ / nt = l$. In a realistic computer vision solution the number of lattices that must be produced by search to find a solution through unification is large due to the many rules and models available in the lattice of feature structures. In this case, the speedup obtained from unifying the search lattices in parallel may be significant. Even in small scale and knowing that the best case speedup is reduced from the ideal value of l, the speedup by unifying the search lattices in parallel is evident. Each search lattice l is explored by a set of nodes and not a single processing node. Therefore, the ideal speedup of l is obtained when using sl processing nodes, where s is the number of processing nodes required to evaluate an average search lattice l.

In this unification in parallel scheme, each processing node would be more complex than the parallel unification scheme. Each node would be required to run the control system to determine a conflict set of rulesets to try for unification. Each processing node would need access to the control system inference engine and meta-rules and be able to run the control using that processing nodes current feature structure representation of the information available in that search lattice currently under investigation by the processing node. In addition, the processing node would need to perform the unification process using the current feature structure and have access to the rulesets to unify against. The processing node would also need the ability to pass its information to a restricted number of nodes within a tree structure to allow the search lattices identified by the applicable rulesets to be explored. Even though the nodes would need only a few connections at a time, these connections would need to be reconfigurable to allow for processing reuse as the solution paths developed. Since the processing nodes need to execute independently, a multiple-instruction multiple-data (MIMD) parallel architecture machine is indicated. Also, many complex processing nodes are required to explore the many search lattices for an image determination. Similar to the parallel unification scheme, current machines exist that meet these processing requirements. Choosing any one of these parallel platforms that allow ease of implementation of the current unification and control software are appropriate for implementation of the unifications in parallel approach.

CHAPTER 7
CONCLUSION

7.1 Discussion

The major contribution from this research is a general and flexible representation technique for model-based computer vision. This representation integrates various sources of knowledge within model-based vision: functional, geometric, and relational. By fully developing this representation, work that is accomplished in functional and other high-level models can be more directly applied in computer vision without the necessity to provide a representational conversion from the image extracted parameters to the represented model parameters.

In the process of achieving the research goal stated above, three significant results were produced. The first result is a complete and systematic approach of relating image description data to the object models was developed and explored. Because the primitive image data and features are directly represented as a feature structure, the feature structure is the common link between the extracted image components and the model components. The use of the unification grammar framework, of which the feature structure is a part, provides a methodology for expressing and manipulating the knowledge required in a model-based computer vision system. The second result is a simplified approach to control and guide the search inherent in this model-based scheme for image analysis. Integrating various control components in an inferencing framework allows the best decisions to be made as to which features structures to unify in the search for the appropriate image analysis. This framework also permits the best of data driven and hypothesis driven algorithms to participate in the solution procedure. The final result deals with parallel approaches for manipulation of the feature structure representation. Possible methods for parallel implementations of the unification grammar approach applied to computer vision were determined. Although the methods were not implemented on a particular platform, a specific benefit was shown for both the parallel unification of feature structures within one search lattice and the ability to unify many search lattices in parallel.

7.2 Future Work

There are additional aspects that have potential relevance to this work that were not in the current research scope. However, these are avenues for expansion in the future. Broad research areas such as image motion and the navigational ability of a computer vision platform may be modelled within the feature structure representation. The feature structure representation for object modelling could also be extended to incorporate scene and object associational information. This extension allows the feature structure representation to be accessed and available to scene interpretation processes. Future enhancement of the operation of the current vision system based upon feature structures is another area of future research. These enhancements would include building of a graphical interface to the lattice of feature structures and using developments from the linguistic community in the techniques to represent feature structures.

7.2.1 Functional Enhancements

The ability to include almost any type of data, function, or image modelling technique is a powerful aspect of the feature structure representation. Specifically, to include the aspect image modelling representation into the feature structure allows the ability to see the change of feature structure determination with each image input into the computer vision system. Then the adjacency information contained by the aspect representation as a feature structure and the relationship of the feature structure resulting from each input image, allow the computer vision system to determine the images as simply different viewpoints of the same object. The change in viewpoint can be determined over time by the connections of the feature structure viewpoint feature and be used to decide image motion parameters. The ability to model this type of viewpoint information into the feature structure representation makes it possible to extend the current computer vision system to include motion identification processes.

Another area of research to include in this type of system is navigational planning. The ability to have processing elements as values of features within the feature structure representation lends itself directly to the incorporation of navigational planning. If the computer vision system is housed in a movable platform, the ability to maneuver that platform through the obstacles identified by the vision system would be an interesting area of development. A significant enhancement for the maneuverable platform would be that the vision system and navigational system are developed using the same computer representation and thus form an integrated computer system.

The incorporation of scene and object associational information would be a significant extension to the system. The modification can be done with relative ease. The work required to incorporate this information would only be in the development of the actual scene and relational information required for representation of the ideas. This type of information can be modeled directly in the feature structure representation. The reasoning power of the computer vision system could be increased dramatically with the addition of this type of information to the current feature structures with no modification of the feature structure representation or manipulation techniques required to use the new information.

7.2.2 Structural Enhancements

The building of a graphical interface tool would be an important step in the actual operation of this computer vision system. This concept is of a visualization tool to allow the viewing of all feature structures within the framework of the lattice and to automatically generate the subsumption relationship among the current system feature structures. The tool would also be an actual interface to the computer vision system. The ability to add, modify, and delete feature structures in the lattice would be needed along with the ability to read in image data directly and put it automatically into the current feature structure representation of that image data. Also within this tool would be an interface to the unification algorithm. As the unification algorithm is executed using the image input, the search lattice would be generated in the visualization tool portion and the system user could directly view the solution path as it is explored by the computer vision system. This type of graphical interface tool would be of primary importance to fielding a computer vision system based upon feature structures and unification. In addition, this tool would also play an important role in each phase of the life-cycle of a fielded computer vision system.

The feature structure representation is a strong area of research in the linguistic community. Because of this linguistic research, many complimentary ideas to this current work are developed concerning the use and functionality of the feature structure representation. One linguistic development allows the feature structure representation to be extendible to a more structured programming framework for future computer vision system applications. These feature structure extensions include typed feature structures and type inheritance hierarchies for the feature structure representation [21, 19]. These areas of feature structure development could prove fruitful in developing a fielded computer vision system.

APPENDICES

Appendix A

Unification Software Documentation

A.1 Purpose of Documentation

The intended audience of this documentation includes users who would like to apply or modify the software implementation of the unification algorithm described in this dissertation. The documentation also provides insight into the implementation for those who desire a more detailed understanding of the unification software than the description presented in the dissertation. The documentation consists of a general description of the unification software and a brief description of each module used in the process of unification.

A.2 General Description

The unification algorithm developed for this dissertation can be used on different levels. The level of use determines the desired input into the unification algorithm. Unification may be accomplished using two feature structures, a feature structure and a list of feature structures, or two lists of feature structures. The module descriptions make it obvious which function required which inputs to the overall unification process. The output of the unification algorithm is a unification feature structure or an indication that the inputs into the unification algorithm did not unify. In the case of failed unification, the original inputs into the unification algorithm are output so that the user may see the unification failure in the inputs. The unification algorithm is implemented in Common LISP.

A.3 Module Descriptions for Unification Algorithm

This section presents a brief description of each module in the implementation. The description contains the purpose of each module, the inputs the module requires, and the outputs or side effects produced by the module.

UNIFY

Description: Unifies the two input feature structures and provides the unified feature structure as output or an indication that unification of the two feature structures is not possible. The unification is order-free (the features of the feature structures may be in any order). The unification software handles the presence of conjunctive, disjunctive, range-valued and multi-valued, and functional processing features.

Input: The two feature structures to unify

Output: A unified feature structure if unification possible or sets the unification feature structure to 'not unifiable' as the indication that unification failed.

Side Effects: The global binding list of shared variables may be altered, if necessary, if unification is successful.

UNIFY-DIS

Description: Unifies two input disjunctive feature structures and provides the unified feature structure as output or an indication that unification of the two disjunctive feature structures is not possible. The disjunctive unification essentially cross-multiplies the (feature value) pairs to combine while checking for any inconsistencies that may cause unification to fail. The items to unify must both be either general disjunctive unification or value disjunctive unification, i.e., the two disjunctive feature structures must be either (feature value) pairs or disjunctive values only. A mix of these two types of features results in the feature structures not unifying.

Input: The two disjunctive feature structures

Output: a unified feature structure if unification possible or the indication that unification failed.

PUT-REMAINING

Description: Inserts the remaining (feature value) pairs of the second unifying feature structure that are not contained in the first feature structure and, therefore, are not contradictory. The global binding list for shared variables is also updated.

Input: The second feature structure in unification, the current unification feature structure, the current binding list

Output: The unified feature structure and the updated global binding list

PUT-REMAINING-DIS

Description: Inserts the remaining (feature value) pairs of the disjunctive feature structure that are not contained in the unified feature structure and, therefore, should be maintained as is in the unified disjunctive feature structure.

Input: The disjunctive feature structure, the current disjunctive unification feature structure

Output: The disjunctive unified feature structure

<u>NOT-UNIFY</u>

Description: Sets the unification feature structure to 'not unifiable'. This happens after a contradiction in the (feature value) pairs of the two input feature structures is identified.

Input: The current unification feature structure

Output: the non-unifiable feature structure

<u>UNIFICATION</u>

Description: Initiates and terminates the unification process of two single input feature structures.

Input: The two feature structures to unify

Output: The unification feature structure or the non-unifiable feature structure

<u>UNIFY-LISTS</u>

Description: Initiates and terminates the unification process of the input image feature structure with sets of rules expressed as feature structures to determine the content of the image.

Input: The input image feature structure

Output: The unification feature structure (determining the content of the image) or a statement that the input feature structure is not unifiable with any model in the lattice

<u>PRINT-UNIFICATION</u>

Description: Outputs the unification feature structure.

Input: The unification feature structure

Output: The printing of the unification feature structure (determining the content of the image) and the global binding list

PRINT-NOT-UNIFICATION

Description: Outputs the unification failure

Input: The input non-unifiable feature structure

Output: The printing of the failure of unification, the two feature structures that did not unify, and the global binding list

DO-EVAL

Description: Provides an interface to execute functions required during unification. The second feature structure did not have this feature and simple processing and returning a value for the (feature value) pair of the feature structure is all that is required.

Input: The input function name

Output: The appropriate function output expressed as a feature structure value as required.

DO-EVAL-W-CHECK

Description: Provides an interface to execute functions required during unification. In addition, this function processes both feature structure values and then determines equality of the values required for successful unification.

Input: The input function name from both feature structures (both functions are processed and the values checked to make sure they match for appropriate feature structure unification).

Output: The appropriate function output expressed as a feature structure value as required or an indication that unification failed.

<u>DO-EVAL-W-CHECK2</u>

Description: Provides an interface to execute functions required during unification. In addition, this function checks the function return value against the second feature structure value required for successful unification.

Input: The input function name and the (feature value) pair to match against for appropriate feature structure unification.

Output: The appropriate function output expressed as a feature structure value as required or an indication that unification failed.

<u>GET-BINDING</u>

Description: Provides the value of a coreference that is contained in a binding list. A coreference in feature structures is not the variable placeholder for future instantiation, but the variable to enforce equality, the value of which is maintained in the binding list.

Input: The coreference number to find the binding value and the current binding list containing the coreference numbers and their values as a list

Output: The binding value or sets to variable if the coreference does not have a current binding

<u>SET-BINDING</u>

Description: Sets the value of a coreference that is contained in a binding list. A coreference in feature structures is not the variable placeholder for future instantiation, but the variable to enforce equality, the value of which is maintained in the binding list.

Input: The coreference number to find the binding value, the current binding list containing the coreference numbers and their values as a list, and the new value of the coreference

Output: The new binding list

SET-BINDING2

Description: Sets the value of a coreference that is contained in a binding list. A coreference in feature structures is not the variable placeholder for future instantiation, but the variable to enforce equality, the value of which is maintained in the binding list. This set-binding is used when both features of the current feature structure are coreferenced and must then be referenced (by equality) to one another.

Input: The coreference numbers for the binding value, the current binding list containing the coreference numbers and their values as a list, and the new value of the coreference

Output: The new binding list

Appendix B

System Control Software Documentation

B.1 Purpose of Documentation

The intended audience of this documentation includes users who would like to apply or modify the software implementation of the control system software described in this dissertation. The documentation also provides insight into the implementation for those who desire a more detailed understanding of the control system software than the description presented in the dissertation. The documentation consists of a general description of the control system software and a brief description of each module used in the control of the process of system unification.

B.2 General Description

The control system software developed for this dissertation is used to control the process of unification in the computer vision system. The control system consisted of a forward chaining inference engine and meta-rules that reasoned about the applicability of feature structure rulesets in the current situation. The control system forms a conflict set consisting of all applicable feature structure rulesets and then chooses the rule-set to unify with based on a variable firing parameter within the vision system. Once a rule-set is identified by the control system, the control system initiates the trial unification of the current feature structure information in the vision system with the rules contained in the rule-set. If any unification attempt with the rule-set is successful, the new system information obtained from the unification with the rules in the rule-set is used to run the control system to find the new rule-set with which to attempt unification. The process of running the control system to find the appropriate rule-set for unification and running the control system to find the appropriate rule-set for unification and the actual unification of the current feature structure information with the chosen rule-set is repeated until the goal identified in the control system is reached (image identification) or the control system runs out of rulesets with which to unify to find a solution. In the latter case, the current system information is presented to the user by the control system with an indication that an actual goal state was not achieved. The control system software and meta-rules are implemented in Common LISP.

B.3 Module Descriptions for Control System Software

This section presents a brief description of each module in the implementation. The description contains the purpose of each module, the inputs the module requires, and the outputs or side effects produced by the module.

FORM-CONFLICT-SET

Description: Forms the conflict set of appropriate rulesets using the meta-rules and the feature structure containing the current image information

Input: The control system meta-rules and the current image feature structure

Output: The conflict set of appropriate rulesets for possible control system firing

MATCH-ANTECEDENTS

Description: Matches the conditions of the meta-rules to the current image feature structure. The conditions of the meta-rule must all be true in the current feature structure for the meta-rule to be applicable.

Input: The current feature structure and the rule antecedents

Output: True if the rule antecedents are contained in the current feature structure, nil otherwise.

GET-RULE-TO-FIRE

Description: Choses the appropriate meta-rule to fire given the control system inputs.

Input: The conflict sets, the conflict resolution strategy, and the meta-rule firing history.

Output: The meta-rule to fire or nil if there are no meta-rules to fire

UNIFY-LATTICE

Description: Implements the overall control of the vision system based upon unification within the lattice structure. This is the loop that processes the meta-rules against the current feature structure information, gets the next meta-rule to fire, fires the meta-rule to get the rule-set for next unification, and initiates the unification cycle of the current feature structure and the rules within the chosen rule-set. The goal condition is checked for unification ending and generalization, if necessary, is also controlled by this function.

Input: The input feature structure to determine image content, the control system meta-rules, and the conflict resolution strategy to use.

Output: The image content result, or an indication that the goal was not obtained and the current image content feature structure.

GOAL-MET

Description: Checks for goal satisfaction in the meta-rule chosen to fire.

Input: The meta-rule chosen to fire.

Output: True if the goal state is indication by the meta-rule, nil otherwise.

CONSEQUENT

Description: Determines the consequent of the chosen meta-rule to fire.

Input: The meta-rule chosen to fire.

Output: The consequent of the chosen meta-rule is the appropriate rule-set for unification.

PRINT-RESULT

Description: Outputs the results after the goal condition is met.

Input: The current feature structure of the image content.

Output: The image content explanation portion of the feature structure and the current system binding list as necessary.

REFERENCES

1. T. O. Binford. Survey of model-based image analysis systems. *The International Journal of Robotics Research*, 1(1):18-64, Spring 1982.

2. R. T. Chin and C. R. Dyer. Model-based recognition in robot vision. *Computing Surveys*, 18(1):67-108, March 1986.

3. A. M. Wallace. A comparison of approaches to high-level interpretation. *Pattern Recognition*, 21(3):241-259, 1988.

4. R. A. Brooks. Symbolic reasoning among 3-D models and 2-D images. *Artificial Intelligence*, 17:285-348, 1981.

5. P. J. Flynn and A. K. Hain. CAD-based computer vision: From CAD models to relational graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(2):114-132, February 1991.

6. W. H. Plantinga. *The ASP: A Continuous, Viewer-Centered Object Representation for Computer Vision.* PhD thesis, University of Wisconsin-Madison, WI, 1988.

7. I. Biederman. Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing*, 32:29-73, 1985.

8. L. C. Topa. *A Rule-Based Expert System for the Determination of Object Structure and Motion Information from a sequence of Digital Images.* PhD thesis, Clemson University, Clemson, SC, 1987.

9. L. Stark and K. Bowyer. Achieving generalized object recognition through reasoning about association of function to structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):1097-1104, October 1991.

10. K. A. Liburdy. *Object Classification in High Level Vision: A Unification Based Approach.* PhD thesis, Clemson University, Clemson, SC, 1991.

11. M. DiManzo, E. Trucco, R. Giunchiglia, and F. Ricci. FUR: Understanding FUnctional Reasoning. *International Journal of Intelligent Systems*, 4:431-457, 1989.

12. J. H. Connell and M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, 31:159-183, 1987.

13. K. Knight. Unification: A multidisciplinary survey. *ACM Computing Surveys*, 21(1):93-122, March 1989.

14. S. M. Shieber. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, Ventura Hall, Stanford University, Stanford, CA, 1986.

15. M. Kay. Parsing in functional unification grammar. In D. R. Dowty, L. Karttunen, and A. M. Zwicky, editors, *Natural Language Parsing: Psychological, Computational, and Theoretical Perspectives*, pages 251-278. Cambridge University Press, New York, NY, 1985.

16. H. B. Laufer. *Discrete Mathematics and Applied Modern Algebra.* PWS Publishers, Boston, MA, 1984.

17. R. J. Schalkoff. *Digital Image Processing and Computer Vision.* John Wiley and Sons, New York, NY, 1989.

18. R. Kasper. Typed feature constraint systems: structures and descriptions. In H. Trost, editor, *Feature Formalisms and Linguistic Ambiguity*, pages 1-19. Ellis Horwood Limited, West Sussex, England, 1993.

19. H. Trost. *Feature Formalisms and Linguistic Ambiguity.* Ellis Horwood, West Sussex, England, 1993.

20. T. Winograd. Frame representations and the declarative/procedural controversy. In R. Brachmand and Levesque, editors, *Readings in Knowledge Representation,* pages 357-370. Morgan Kaufmann Publishers, San Mateo, CA, 1985.

21. B. Carpenter. *The Logic of Typed Feature Structures.* Cambridge University Press, New York, NY, 1992.

22. F. Quek, R. Jain, and T. E. Weymouth. An abstraction-based approach to 3-D pose determination from range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(7):722-736, July 1993.

23. P. G. Frankl and W. J. Weyuker. An applicable family of data flow testing criteria. *IEEE Transactions on Software Engineering*, 14(10):1483-1498, October 1988.

24. L. A. Clarke, A. Podgurski, D. J. Richardson, and S. J. Zeil. A comparison of data flow path selection criteria. In *Proceedings of the Eighth International Conference on Software Engineering*, pages 244-251, IEEE Computer Society, London, UK, August 1985.

25. C. J. Thornton and B. Du Boulay. *Artificial Intelligence Through Search.* Kluwer Academic Publishers, Norwell, MA, 1992.

26. D. Partridge. *A New Guide to Artificial Intelligence.* Ablex Publishing Corporation, Norwood, NJ, 1991.

27. R. J. Schalkoff. *Artificial Intelligence: An Engineering Approach*. McGraw-Hill, Hightstown, NJ, 1990.

28. M. Emele. Unification with lazy non-redundant copying. In *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*, pages 325-330, ACL, University of California, Berkeley, CA, USA, 1991.

29. K. Kogure. Types feature structure generalization by incremental graph copying. In H. Trost, editor, *Feature Formalisms and Linguistic Ambiguity*, pages 141-158. Ellis Horwood Limited, West Sussex, England, 1993.

30. J. Matiasek. Structure sharing unification of disjunctive feature descriptions. In H. Trost, editor, *Feature Formalisms and Linguistic Ambiguity*, pages 93-102. Ellis Horwood Limited, West Sussex, England, 1993.

31. A. C. Shaw. A formal picture description as a basis for picture processing systems. *Information and Control,* 14:9-52, 1969.

32. K. Hasida. Dynamics and parallel disjunctive unification. In H. Trost, editor, *Feature Formalisms and Linguistic Ambiguity*, pages 79-92. Ellis Horwood Limited, West Sussex, England, 1993.